ANNO 1 - N. 2 SETTEMBRE/OTTOBRE 1988 L. 14.000 - Frs. 21.00

AMIGA MAGAZINE

GRUPPO EDITORIALE JACKSON
DIVISIONE PERIODICI

PASSE-PARTOUT NELL'UNIVERSO DI AMIGA

CONTIENE: AMIGA Disk Magazine

Xerox 4020

Curve Spline

True Basic

Forth e musica

Spedizione in abbonamento postale Gruppo III/70 Taxe percue (Tassa riscossa) Milano Ferr. Corr.

ENCICLOPEDIA PRATICA JACKSON DI ELETTRICITÀ & ENERGIA

IN EDICOLA 2 FASCICOLI A SOLE 200 LIRE

In Regalo
agli acquirenti dell'opera
un favoloso telefono a tastiera con memoria

ENCICLOPEDIA PRATICA JACKSON DI ELETTRICITÀ & ENERGIA 1
IMPIANTI E MATERIALI
FAI DA TE
BASI DELL'ELETTRICITA'
ELETTRODOMESTICI
FONTI DI ENERGIA

ENCICLOPEDIA PRATICA JACKSON DI ELETTRICITÀ & ENERGIA 2
IMPIANTI E MATERIALI
FAI DA TE
BASI DELL'ELETTRICITA'
ELETTRODOMESTICI
FONTI DI ENERGIA

52 fascicoli da rilegare in:
4 splendidi volumi
con un totale di 1050 pagine
oltre 5000 fotografie e illustrazioni

IN OGNI FASCICOLO

- FAI DA TE
- BASI DELL'ELETTRONICA
- ELETTRODOMESTICI
- IMPIANTI E MATERIALI
- FONTI DI ENERGIA

ELETTRICITÀ & ENERGIA è la grande opera del Gruppo Editoriale Jackson nata per tutti coloro che intendono acquisire la padronanza più completa delle fonti energetiche, dalle tecnologie utilizzate, fino alle principali applicazioni.

Grande spazio è dedicato all'*elettricità*, dalle sue leggi fondamentali, fino ai suoi più comuni settori di utilizzo. L'elettricità è, infatti, tra tutte le risorse energetiche, quella, con cui chiunque di noi ha quotidianamente a che fare.

Rivolta all'hobbista oltre che al tecnico, ELETTRICITÀ & ENERGIA riserva un buon numero di pagine, in ogni fascicolo, anche a nozioni di tipo pratico, dall'impiantistica al "fai da te" elettrico.

Tutti gli argomenti sono trattati con lo stile e la professionalità delle Grandi Opere Jackson.

ENCICLOPEDIA PRATICA JACKSON DI

IL SAPER FARE DI OGNI GIORNO

E siamo già al secondo appuntamento.
Forse più familiari; credibili nei nostri sforzi.
E così, in questo numero, si riconfermano, insostituibili, i tutorial ai diversi linguaggi di Amiga: AmigaBasic, C, Assembly; e al CLI. E si ripresenta, sempre smagliante e sorprendente, Disco Magazine.
Ma ci sono pure delle novità. Vi introduciamo a True Basic, un'implementazione del Basic che offre nuove possibilità ai cultori di questo linguaggio. E sempre in Basic c'è ROT, uno straordinario programma grafico che vi permette di progettare e muovere figure tridimensionali.
E tra le altre cose, un interessante articolo sulla stampante Xerox 4020, le cui stampe già avete potuto ammirare sulle pagine del primo numero della rivista e che continuerete ad ammirare anche in questo. Un modo insomma per presentarvi più da vicino questo nostro formidabile collaboratore.
E poi c'è Forth.
La sua presenza, è una precisa scelta in direzione di una informazione il più possibile completa sulle implementazioni dei diversi linguaggi per Amiga. E qui ci fermiamo per alcune considerazioni. È mai possibile che non si possano rendere disponibili sul mercato italiano le molte implementazioni dei diversi linguaggi per Amiga, affermate ormai da tempo oltreoceano?! Queste difficoltà non solo intralciano il nostro lavoro, ma rischiano di rallentare la diffusione dell' Amiga e lo sviluppo di software adeguato alle possibilità della macchina. E qui non si vuole solamente lamentare l'irreperibilità sul mercato ufficiale di validi prodotti come Multi-Forth della Creative Solutions e JForth della Delta Research, ma di implementazioni ormai quasi ' leggendarie ' di Lattice, Manx e Metacomco per esempio, per non parlare poi di tutti gli altri validi prodotti di cui molto spesso non si conosce nemmeno l'esistenza.
Non potremmo a questo punto che esclamare anche noi: pirata e' bello?!
Il nostro compito, infatti, è anche quello di penetrare in questo lato nascosto e oscuro dell'universo Amiga e farlo parlare: it's black.

La Redazione

SOMMARIO

Foto di copertina tratta da:
«Shangai»

Anno I numero 2 Settembre/Ottobre 1988

DIRETTORE RESPONSABILE
Giampietro Zanga

REDAZIONE
Graphic & Comp. Gorizia

COORDINAMENTO REDAZIONALE
Simone Concina

ART DIRECTOR
Gianni Marega

COLLABORATORI
Roberto Beccia
Primoz Beltram
Tomi Beltram
Fabio Biancotto
Giorgio Dose
Mr. Lambda
Massimo Lavarin
Furio Lusnig
Luigi Manzo
Giovanni Michelon
Emilio Orione
Alessandro Prandi
Giacomo Pueroni
Paolo Russo

GRAFICA, IMPAGINAZIONE, COPERTINA
Graphic & Comp.

DIVISIONE PUBBLICITÀ
Via Pola, 9 - 20124 MILANO - Tel. 69.481
Telex 316213 REINAI - 333436 GEJ - ITI
OVERSEAS DEPARTMENT: Tel. 02/6948201
PUBBLICITÀ PER ROMA-LAZIO E CENTRO SUD
Via Lago di Tana, 16 - 00199 Roma
Tel. (06) 8380547 - Telefax (06) 8380637

FOTOCOMPOSIZIONE
FOTOFORMA - Via del Molino a Vento, 72
34137 TRIESTE

STAMPA
Grafika, 78 - Pioltello

DISTRIBUZIONE
Sodip - Via Zuretti, 25 - 20125 MILANO
Spedizione in abbonamento postale Gruppo III/70
Pubblicità inferiore al 70%

UFFICIO ABBONAMENTI
Tel. (02) 6122527-6187376
Prezzo della rivista L. 14.000 (Frs.21.00)
Numero arretrato L. 28.000
Abbonamento annuo L. 120.000
per l'Estero L. 270.000
I versamenti vanno indirizzati a:
Gruppo Editoriale Jackson
Via Rosellini, 12 - 20124 Milano
mediante emissione di assegno bancario, vaglia o utilizzando il C/C/ postale numero 11666203
Per i cambi di indirizzo, indicare, oltre al nuovo, anche l'indirizzo precedente, ed allegare L.500, anche in francobolli.

DIREZIONE, REDAZIONE, AMMINISTRAZIONE
Via Rossellini, 12 - 20124 Milano
Tel. (02) 68.80.951/2/3/4/5 - Telex 333436 GEJIT I

SEDE LEGALE
Via G. Pozzone, 5 - 20121 Milano

Il Gruppo Editoriale Jackson è iscritto nel Registro nazionale della Stampa al n. 117 vol. 2 - foglio 129 in data 17/8/1982

Autorizzazione del Tribunale di Milano n. 102 del 22/2/1988

Prowrite

È stata introdotta di recente sul mercato una versione aggiornata del ProWrite V2.0 con molte nuove configurazioni. Di particolare interesse è la possibilità di selezionare le specificazioni di default. Tutti i parametri di avvio come font, giustificazioni, spaziatura e colori possono essere predeterminati e vari file preference possono venir immagazzinati e caricati in memoria all'occorrenza.

Anche la memoria di lavoro riservata alla grafica e al testo è stata utilizzata al meglio, migliorando la velocità che ora è di dieci volte più rapida della versione precedente di questo programma e nel modo testo tale velocità è veramente sorprendente. Le stampanti che prevedono l'output in alta risoluzione permettono l'output di true letter quality da qualsiasi font di Amiga. Gli elaborati possono essere di qualsiasi larghezza utilizzando l'opzione del programma che permette la stampa su fogli di qualsiasi larghezza.

I disegni HAM, tipo quelli creati con DigiPaint, possono essere caricati all'interno del ProWrite e utilizzati. Il paragrafo riguardante le opzioni di formattazione del testo permette di inserire spazi interi o dimezzati, fissare l'altezza della linea e l'inclusione di linee vuote di separazione tra i vari paragrafi. I paragrafi finali possono essere ordinati in modo ascendente o discendente per permettere la creazione di tabelle o per editare il dizionario che è compreso nel programma. Tale dizionario può verificare l'esatta grafia di 95.000 parole. Il conteggio di caratteri, parole, frasi, paragrafi e di facile utilizzo riuscendo così ad ottenere in modo semplice tutte le informazioni riguardanti il documento. Queste sono alcune delle principali migliorie rispetto alla versione precedente del programma, ma ce ne sono molte altre che contribuiscono al potenziamento di questa versione.

Il prodotto è fornito dalla New Horizons Software, P.O. Box 43167, Austin, Texas, 78745, USA.

Hard News

Una nuova serie di sistemi hard disk compatibili per Amiga 500 è stata immessa sul mercato statunitense dalla Supra Corporation. L'hard disk drive è disponibile con quattro diverse capacità, con porta d'espansione SCSI e fornisce inoltre delle capacitàdi espansione della RAM. I quattro drive hanno una capacità di 20, 30, 60 e 250 Mbyte e sono disponibili, rispettivamente al prezzo di $995, $1195, $1995, e $3995. I Supra Drive vanno inseriti direttamente nel socket d'espansione dell'Amiga 500 ed il trasferimento dei dati è molto rapido. RAM board di 1 o 2 Mbyte di capacità possono essere innestate nell'unitàdell'hard disk.

Per maggiori informazioni contattate: Supra Corporation, 1133 Commercial Way, Albany, OR 97321, USA.

Accordo tra G.E. Jackson e VNU Business Press

Il Gruppo Editoriale Jackson S.p.A., editore leader in Italia nel settore dell'elettronica, dell'informatica e delle nuove tecnologie, e la VNU Business Press Group, gruppo internazionale con sede ad Amsterdam e consociate negli USA, Inghilterra, Francia, Spagna, Belgio e Australia, annunciano di aver concluso un accordo che prevede l'ingresso del gruppo olandese nella compagine azionaria del G.E. Jackson.

L'accordo risponde in modo preciso alla struttura del mercato delle tecnologie avanzate, il quale evidenzia la necessità di superare i confini nazionali. L'accordo prevede la diffusione tempestiva su scala internazionale dei nuovi media interattivi realizzati dalla Jackson, che vanta un'esperienza unica nel settore dei prodotti autodidattici basati su personal computer.

Jackson e VNU svilupperanno anche tutte le sinergie possibili nell'area dei nuovi media video, telematici e ottici con particolare riferimento alle tecnologie CD-ROM.

La VNU è nota nel settore dell'editoria elettronica internazionale, oltre che per l'incorporazione dell'autorevole casa editrice americana Hayden Publishing Company (Electronic Design), anche per la recente acquisizione della prestigiosa rivista Electronics della McGraw-Hills. Nel settore dell'informatica la VNU pubblica testate di rinomanza mondiale quali l'americana Personal Computing, le inglesi Personal Computer World e Computing, le francesi Informatique Hebdo e Soft & Micro e la spagnola Chip.

Attualmente la Jackson pubblica 28 riviste, oltre 800 manuali tecnici e scolastici, 16 tra enciclopedie e corsi interattivi di autoapprendimento.

Sim-Hi.Fi-Ives ventiduesima edizione

La ventiduesima edizione del Salone Internazionale della Musica e High Fidelity - International Video and Consumer Electronics Show si svolgerà nei Padiglioni 7, 12, 13, 14, 14B della Fiera di Milano dall'8 al 12 settembre 1988.

Il SIM-HI.FI-IVES di Milano rappresenta una delle principali manifestazioni specializzate del proprio settore e rappresenta una impagabile vetrina sulle anticipazioni e novità dei produttori oltre ad essere una grandiosa occasione di animazione del mercato nazionale e internazionale.

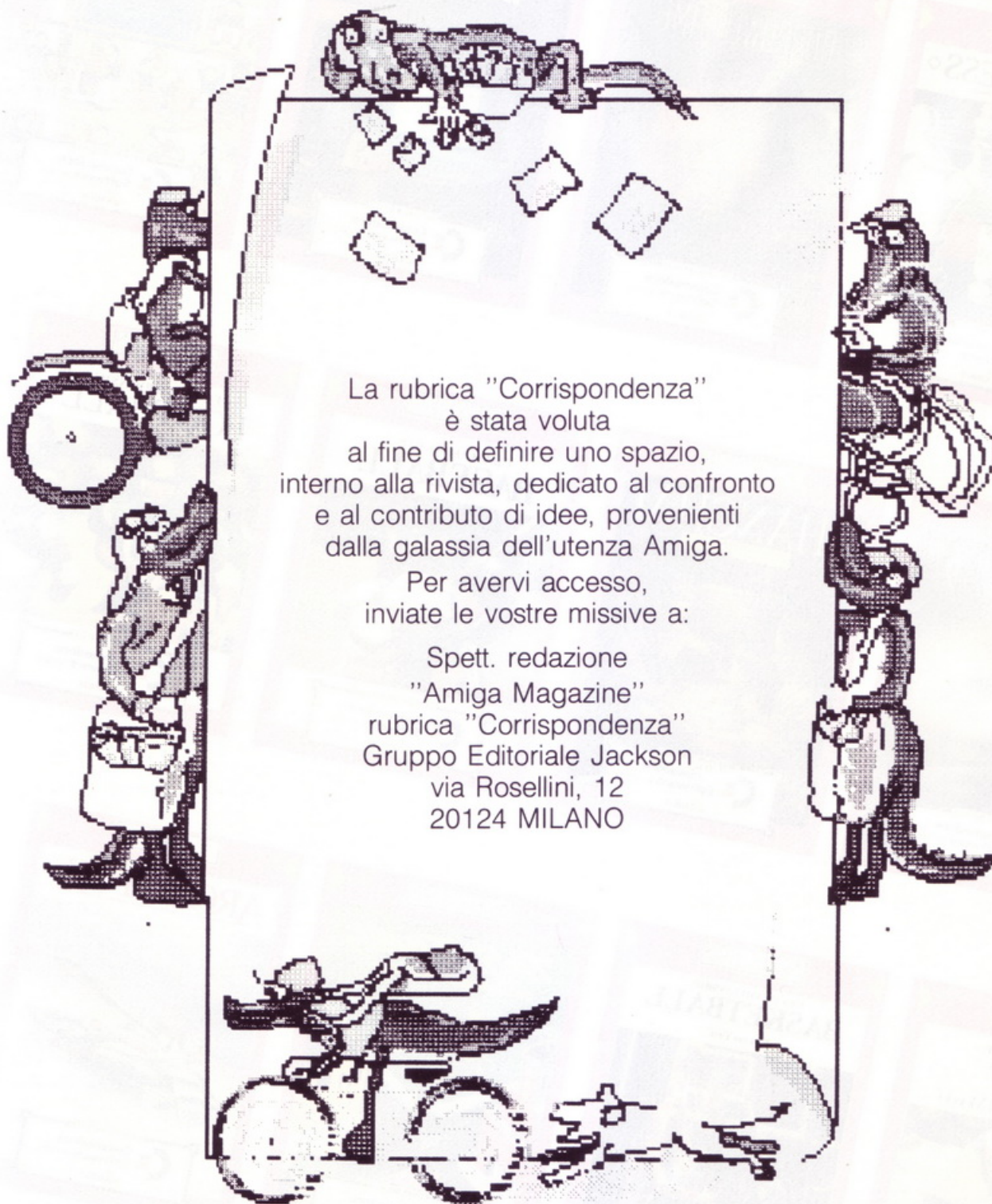
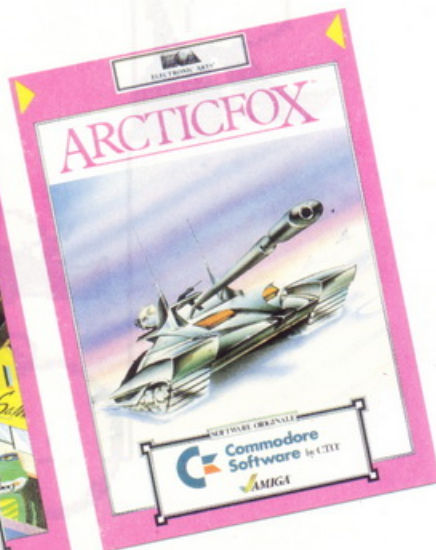
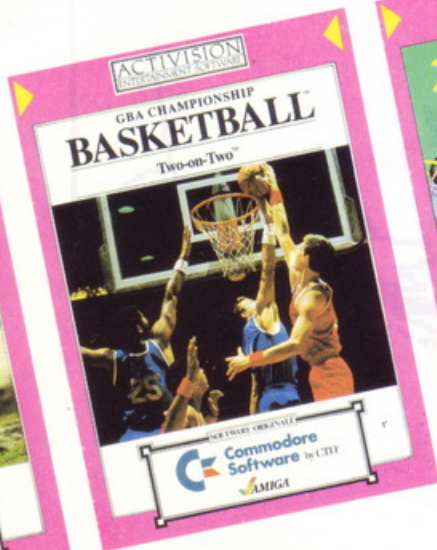
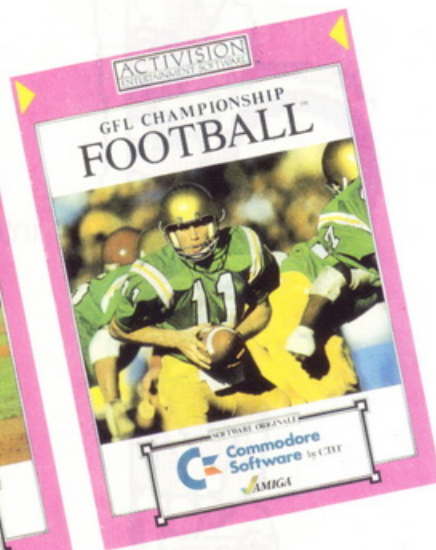
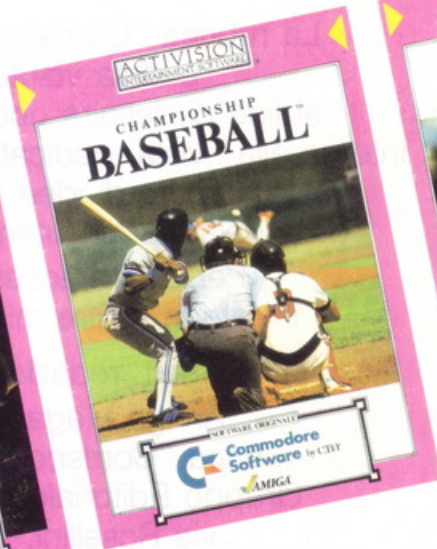
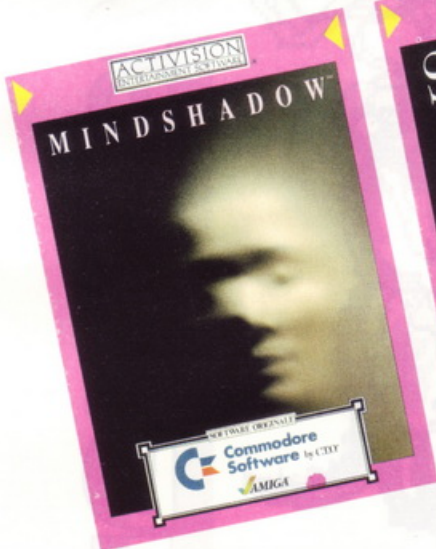
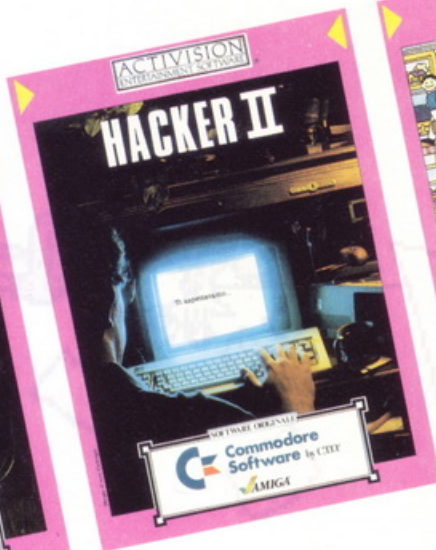
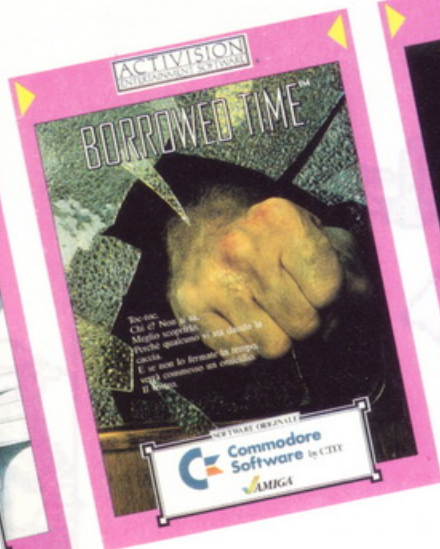
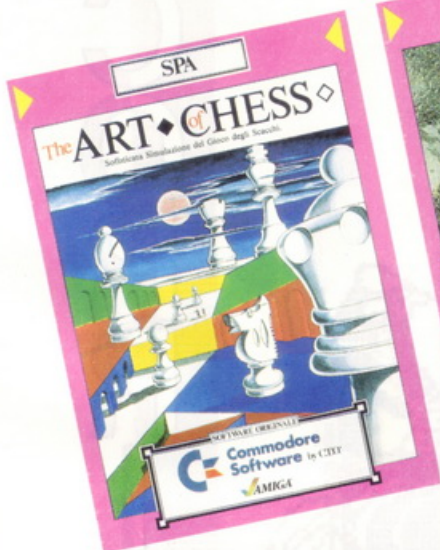
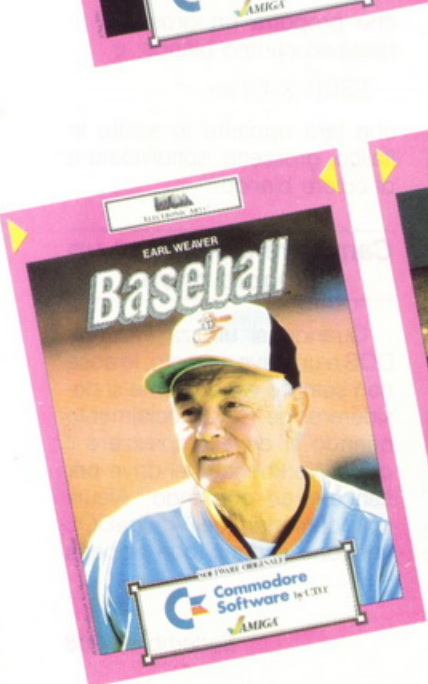
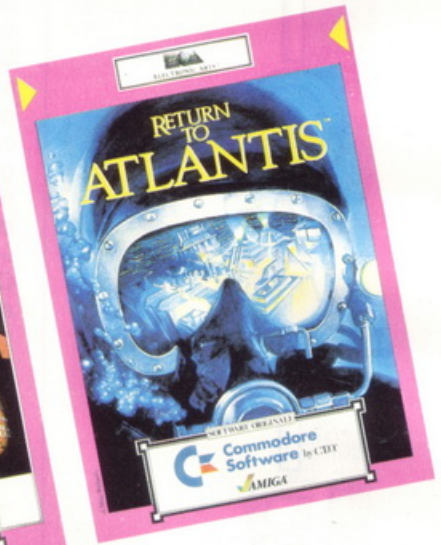
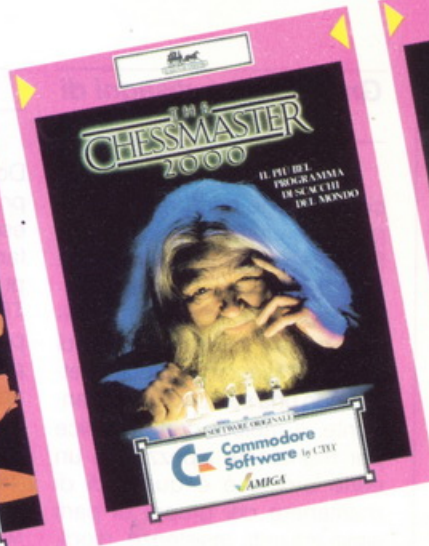
La superficie prevista per questa edizione è di 35.000 metri quadrati. Sono previsti 300 espositori diretti per un totale di oltre 800 ditte rappresentate.

Dall'8 al 12 settembre il SIM-HI.FI-IVES sarà l'unica manifestazione espositiva della Fiera di Milano, perciò per quanto riguarda i parcheggi, i posti-letto, le prenotazioni aeree, Milano sarà a completa disposizione degli operatori specializzati presenti.

CORRISPONDENZA

La rubrica "Corrispondenza"
è stata voluta
al fine di definire uno spazio,
interno alla rivista, dedicato al confronto
e al contributo di idee, provenienti
dalla galassia dell'utenza Amiga.

Per avervi accesso,
inviate le vostre missive a:

Spett. redazione
"Amiga Magazine"
rubrica "Corrispondenza"
Gruppo Editoriale Jackson
via Rosellini, 12
20124 MILANO

AMIGA TRICKS

Graphicraft e disegni di circuiti

Operando con il Graphicraft, ci si accorge ben presto che è possibile effettuare l'editing dei file Brush. Una volta editati tali file è possibile modificarli in simboli elettrici e registrarli su disco; questo fatto permetterà di vcreare disegni di circuiti elettrici con una semplicità sorprendente. Il metodo più semplice per mantenere i pezzi con un certo ordine è è quello di di mantenere dei gruppi di parti simili (cavetti, resistenze, diodi, ecc.) sulla stessa palette brush. Spostarsi tra i file brush non è affatto difficile, ma fa perdere del tempo. Unico consiglio che crediamo utile fornirvi è: Disegnate i componenti orientati nelle quattro direzioni, in questo modo potrete sistemarli sui vostri disegni più facilmente. Questo metodo è meno espensivo di quello proposto da vari programmi commerciali ed ha un unico limite che è quello determinato dalla vostra immaginazione.

Say

Digitando il Comando SAY dal CLI si scoprono alcune interessanti configurazioni di questo comando. Appena digitato il comando SAY, apparirà sullo schermo una window che illustrerà alcuni degli utilizzi di questo comando. Anche se gli esempi sono abbastanza vaghi, sono sufficienti per indirizzarci nella direzione giusta. Vi proponiamo la sintassi del comando ed un esempio di come l'Amiga legge i file testo per voi:

Sintassi:
SAY-X (dir)/(filename[.ext])
Esempio:
SAY-X S/startup-sequence

È evidente che quanto proposto non servirà solamente per divertirsi nell'ascoltare la macchina mentre svolge questa funzione, ma piuttosto potrà essere utilizzato ogni volta che si desidera ascoltare un testo scritto in precedenza.

Comandi AmigaDos

Molti comandi dell'AmigaDos generano dei prompt propri mentre attendono l'input. Ad esempio, il comando DATE? farà apparire il seguente prompt sullo schermo: TIME,DATE,TO = VER/K. È possibile però eliminare il comando prompt utilizzando l'operatore ridirezionale > e il device NIL. In questo modo DATE>NIL:? invierà il prompt al computer 'nella terra di nessuno', questo, senza generare degli errori. Assicuratevi solamente di posizionare l'operatore ridirezionale prima del parametro richiesto.

Maggior spazio sui dischetti

Un modo per ottenere maggior spazio sui dischetti, è quello di cancellare i file che non interessano nella directory SYS:DEVS/PRINTERS. Se utilizzate uno dei printer driver forniti dalla Preference, dovrete semplicemente cancellare gli altri che non utilizzerete. Per ottenere quanto appena proposto digitate quanto segue dal prompt del CLI:

```
CD DEVS
COPY PRINTERS/GENERIC
TO SYS:TCD :
DELETE SYS:DEVS/PRINTERS
ALL CD DEVS
MAKEDIR PRINTERS
COPY SYS:T/GENERIC TO
PRINTERS CD :
```

È una buona idea comunque, il richiedere un DIR DEVS/PRINTERS prima di procedere con la cancellazione dei file printer, semplicemente per verificare come viene listata la vostra stampante. Non effettuate quanto proposto sul disco originale del Workbench, ricordatevi che in un futuro anche non lontano potrete decidere di acquistare una nuova stampante!

Modi testo e CLI

Dal CLI è possibile abilitare i vari modi testo con poche e semplici pressioni di tasti. La sequenza per ottenere quanto detto è molto semplice:

ESC[n1;n2;n3m

dove ESC è la escape key, 'n1' e' il numero dello stile, 'n2' e' il numero per il colore del primo piano, 'n3' è il colore dello sfondo e 'm' e' la sequenza terminator. Di seguito vi proponiamo l'elenco dei valori che potete utilizzare, ma ricordatevi che se i colori sono stati modificati per mezzo della Preference, questi numeri daranno risultati differenti.

STILE
0 = Testo Normale
1 = Grossetto
3 = Italico
4 = Sottolineato
7 = Inverso
PRIMO PIANO
30 = Default
31 = Bianco
32 = colore complementario binario
(colore di default è nero)
33 = Rosso
SFONDO
40 = Default
41 = Bianco
42 = colore complementario binario
43 = rosso

È possibile naturalmente combinare tra loro i vari stili, immettendo ciascun numero di stile separato da un punto e virgola. Ad esempio:

ESC[1;33;41m

che proporrà un grossetto in rosso su campo bianco; e

ESC[1;3;4;31m

che farà apparire le scritte in italico, grossetto, sottolineato e di colore bianco.

Cambiare dischi con un singolo drive

Quando si utilizza l'AmigaDOS e un drive soltanto, le cose non sempre vanno come si desidererebbe. Specialmente quando si deve rimpiazzare il disco Workbench nel drive per caricare un comando. Alcuni dei comandi dell'AmigaDOS vengono eseguiti non appena

caricati in memoria, non permettendo così di immettere il disco senza Workbench con cui si intende operare nel drive. Utilizzando il comando CD questi non lavora perché, una volta caricato dal disco Workbench, viene eseguito istantaneamente. Un metodo per aggirare questo problema è il seguente: quando si imposta uno dei comandi ad esecuzione immediata basterà aggiungere uno spazio e un punto di domanda prima di premere il tasto Return, attivando in questo modo l'help on-line dell'AmigaDOS. Si otterrà in questo modo la sintassi ed il formato di ogni comando dell'AmigaDOS (per i principianti questa è una procedura da ricordare) e carica in memoria il comando prima di eseguirlo. Questa tecnica è valida per ogni comando dell'AmigaDOS anche se questi sono privi di parametri, come il comando INFO. Digitate semplicemente INFO? e vi apparirà il messaggio none: Inserite il disco del quale intendete avere delle informazioni e premete il tasto Return. Certamente la conoscenza di questa procedura semplificherà i problemi per tutti coloro che sono in possesso di un drive singolo.

Deluxe Paint nascosti dall'area Undo

Nel disegnare nelle aree dello schermo che sono normalmente coperte dai menù a barre (in alto e sulla destra), non è possibile utilizzare l'opzione UNDO per correggere gli eventuali errori commessi (probabilmente perché riattivando il menù a barre viene considerato come il selezionare qualcosa di nuovo). Un modo per aggirare questo inconveniente è quello di utilizzare l'opzione Center Picture. Si sposti semplicemente il cursore sulla destra o in alto sullo schermo e si prema il tasto 'N', che effettuera' uno scroll automatico della figura, in questo modo le parti del disegno coperte dai menù verranno liberate. Ora potrete modificarle a vostro piacimento.

Multi preference

Se il Preference è un ottimo tool per costruire gli ambienti Amiga, è comunque limitato dal fatto che è possibile registrare solamente un set di sistema di preference mentre se ne utilizza un altro. Coloro che utilizzano più di una stampante, o desiderano un controllo più preciso del loro mouse per il disegno al posto del normale e rapido mouse utilizzato per spostarsi velocemente sullo schermo probabilmente sono interessati ad un modo automatico per spostarsi tra sistemi di preference specifici. Per dare una risposta a questo tipo di problema, provate ad utilizzare i seguenti due comandiche troveranno posto nella vostra SYS:directory. Il primo file:

```
file save-sys
KEY name/a
copy
  SYS:devs/system-configuration
  to
  SYS:devs/system-<name>
end of file
```

Il secondo file:

```
KEY name
IF EXISTS
  sys:devs/system-<name$normal>
  copy SYS:devs/system-<name
  $normal> to
  SYS:devs/system-configuration
  SYS:preferences
ELSE
ECHO "SYS:devs/system-<name
  $normal>
  not found..."
ENDIF
end of file
```

La prima operazione da compiersi è EXECUTE SAVE-SYS NORMAL, che produce una copia dell'attuale configurazione di sistema registrato. Questa dovrà essere la configurazione che pesate di utilizzare più spesso. È possibile effettuare il restore di questa configurazione per mezzo di EXECUTE RESTORE-SYS, senza dover aggiungere alcun argomento. Quando viene visualizzato lo schermo Preference, dovrete semplicemente premere il pulsante (click) su LAST SAVED e USE. Si potrà effettuare un SAVE o RESTORE delle altre configurazioni (es. per una seconda stampante) nel seguente modo:

```
EXECUTE SAVE-SYS PRINTER2
EXECUTE RESTORE-SYS
PRINTER2 (click LAST SAVED e
USE)
```

EXECUTE in S

Se utilizzate sovente i file Execute dell'AmigaDOS, sarete sorpresi di apprendere che un file Execute, posto all'interno della directory S sul Workbench, sarà comunque accessibile indipendentemente dalla directory in uso. Potrete spostare un file alla directory 'Sequence' per mezzo di COPY file TO S:. Troverete utile effettuare una seconda copia del comando Execute in modo abbreviato. Ad esempio: COPY C:Execute to C:X.

Spazi nei titoli Window

Se possedete solamente il manuale Introduction to Amiga e vi siete preposti il compito di introdurre uno spazio nel titolo di una window è possibile che incappiate in parecchi errori prima di approdare ad una soluzione soddisfacente. Comunque esiste un metodo per aggirare l'ostacolo basterà utilizzare l'ALT e la barra spaziatrice (o CTRL-N se siete nel set di caratteri alternato). Tutto quello che dovrete fare sarà di mantenere premuto il tasto ALT e nel frattempo premere la barra spaziatrice, e finalmente l'Amiga accetterà il comando.

Ask per amiga 500 e 2000

Questo è un modo estremamente semplice per condizionare l'installamento di comandi CLI nella RAM disk e poi assegnar loro, gli appropriati volumi virtuali. Incluso nel disco Workbench dell'Amiga 500 e 2000 c'è un comando chiamato ASK. Tale comando vi permette di effettuare un entry in un batch file (in questo caso, Startup-Sequence) che invia un prompt all'utente per una risposta del tipo 'yes or no'. La sintassi per il comando ASK è:

```
ASK "Any text question in
QUOTES"
```

Poi la risposta viene analizzata per mezzo di un comando IF WARN (WARN è vero se la prima lettera della risposta dell'utente è "Y" o "y"):

```
IF WARN
(operazione)
ENDIF
```

Eccovi un esempio per installare la directory in C nella RAM:

```
Echo ""
ASK "Installo i comandi CLI
nella RAM:Drive (Y o N)?"
  IF WARN
      MAKEDIR RAM:C
      COPY C: ALL RAM:C
      QUIET
      ASSIGN C: RAM:C
  ENDIF
```

Poi prima della linea ENDCLI>NIL:, si immetta quanto segue:

```
  IF EXISTS RAM:C
    NEWCLI
CON:540/150/100/50/CLI
  ENDIF
```

Questo permette di decidere se operare solamente in ambiente Workbench o creare un CLI per utilizzarlo con questi comandi. Se i comandi sono stati copiati nella RAM: è certo che si vuole utilizzare un ambiente CLI, mentre l'effettuare un RAM:C presume che si voglia operare solamente con il Workbench. L'unico neo nell'utilizzo della configurazione RAM:C è il rallentamento dell'operazione di booting e la perdita di circa 199K di RAM.

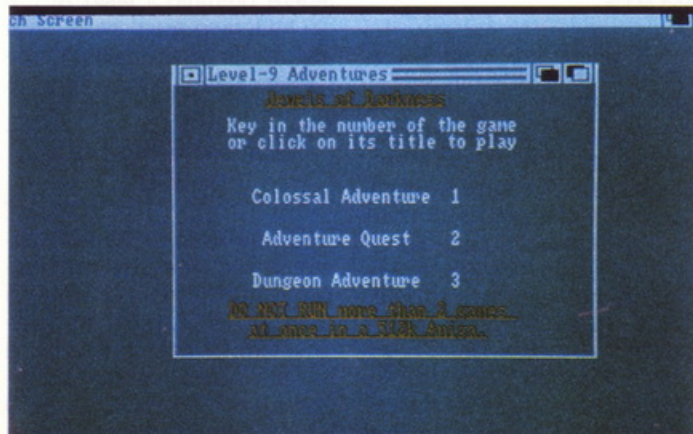
Shanghai

... la concentrazione è spesso la chiave del successo!

Nel corso dei secoli vari giochi sono stati creati; ma solo i migliori tra questi giochi hanno resistito al logorio del tempo e sono giunti sino a noi. Un fatto li accomuna, ed è che sono costruiti su basi molto semplici, ma per raggiungere il traguardo richiesto da ogni singolo gioco, le strategie da utilizzarsi sono spesso incredibilmente complesse. Citando alcuni esempi di questi giochi, potremmo ricordare gli Scacchi, la Dama, il Go (oggi noto anche con il nome di Otello), ed infine i giochi di carte (pensate solamente a quante varianti ci sono). Shanghai è uno di questi giochi ed entra a pieno diritto nella lista dei Giochi Classici.

L'origine di questo gioco si trova in uno dei più antichi giochi del mondo e al tempo stesso più misteriosi. Si dice che risalga a 25 secoli fa circa, ma di sicuro possiamo affermare che la sua patria è la Cina. Questo gioco prende il nome di Mah-Jong, agli inizi veniva giocato solo a corte o al massimo fra cerchie ristrettissime di dignitari del Celeste Impero. Ha avuto nel tempo numerosissime derivazioni (Shanghai è una di queste), ed è uno dei giochi più diffusi nel mondo. Nella sua patria d'origine viene considerato il gioco nazionale per eccellenza.

Le 144 tessere di cui è composto il gioco sono, nella versione originale, fatte di osso o di bambù. Pescatori e marinai lo preferivano e lo preferiscono tuttora ad altri passatempi durante le interminabili traversate oceaniche. La prima esportazione del gioco verso il Giappone (1905-1910) venne fatta da alcuni militari giapponesi. Verso il 1920 alcuni marinai americani importarono da Macao negli Stati Uniti alcune confezioni di questo gioco. In Europa arrivò solamente nel 1922, per mezzo di ambulanti cinesi espatriati in cerca di fortuna.

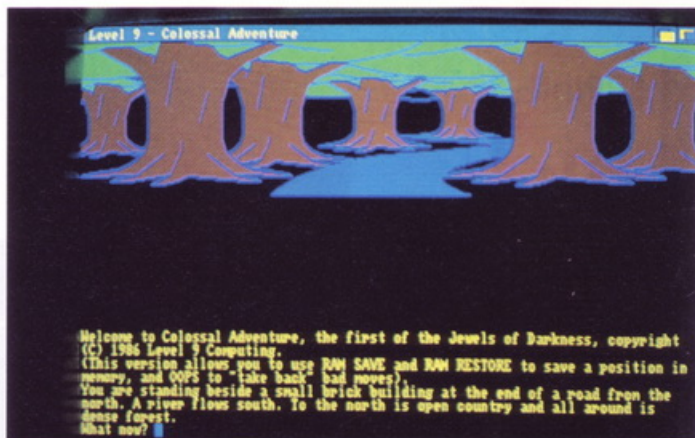
Shanghai è fornito di poche ma essenziali opzioni che dovrete cercare di ricordare. Il gioco richiede l'utilizzo di un sistema che sia dotato di un minimo di 512K. Comunque, se il vostro Amiga ha più di 512K ed è fornito di doppio drive, è possibile avere il Shanghai caricato simultaneamente con un secondo programma. Per giocare è richiesto l'utilizzo del mouse; la tastiera viene utilizzata per immettere il nome dei vincitori o per registrare una partita. Nella modalità Challenge (sfida tra due giocatori), sarà necessario l'inserimento di un secondo mouse nella porta due per servire simultaneamente i due contendenti. Questa opzione è molto gradita da coloro a cui non piace spostare continuamente i controlli. Shanghai non deve il suo successo alla grafica, ma la grafica della versione dell'Amiga è superba. Quello che vi manterrà incollati al monitor

sarà però la strategia di questo gioco (prima di iniziare questo articolo abbiamo speso due pomeriggi interi a provarlo!!!).

Il gioco consiste nel togliere coppie identiche di pezzi dalla configurazione di 144 pedine poste una sopra l'altra come dalla figura che vi apparirà sullo schermo. Questa piramide di pezzi prende il nome di Formazione Drago, dal momento che dovrebbe rappresentare un Drago dormiente, l'effetto tridimensionale è veramente efficace. I pezzi formano al centro una pila di cinque tessere che decresce progressivamente verso l'esterno fino a raggiungere sui bordi delle pile formate da un pezzo soltanto. Per distinguere i livelli da dove intendete togliere i pezzi, le pedine sono provviste d'ombra e sono in questo modo evidenziate. Il livello più in alto proietta la sua ombra su quello immediatamente inferiore. Pare quasi che sia possibile toccare i diversi livelli. Lo sfondo su cui si stagliano le pedine è un velluto di colore verde che offre al gioco un aspetto classico, come fosse giocato su un tavolo ricoperto da un panno. Queste schermate grafiche offrono l'opportunità di apprezzare la potenzialità dell'Amiga.

I 144 pezzi, a forma di domino, sono divisi in vari gruppi. Esistono fondamentalmente due tipi diversi di pedine: quelle numerali e quelle letterali. Le prime sono 104 e a loro volta si suddividono in tre serie (o semi): personaggi (o caratteri), cerchi e bambù (o canne). Ogni seme è composto da 36 pezzi: una serie completa dall'uno al nove è ripetuta quattro volte (es. quattro 3 di bambù, quattro 9 di cerchi, quattro 1 di personaggi, ecc.).

Anche le rimanenti 40 pedine letterali si dividono a loro volta in varie serie: venti, draghi, fiori e stagioni.

I venti sono in tutto sedici, quattro per ogni punto cardinale principale. I draghi sono dodici, quattro per serie (draghi rossi, verdi e bianchi). Gli ultimi otto pezzi saranno ripartiti in misura uguale tra fiori e stagioni. I pezzi sono disegnati fin nei minimi dettagli in maniera impeccabile. Le pedine numerali oltre al disegno hanno sulla parte superiore a destra il numero corrispondente in cifre arabe. I Draghi, i Venti, le Stagioni, ecc., possono essere considerati gli equivalenti dei Re, Regine e Fanti che si trovano nel gioco delle carte. Se consideriamo la longevità del Mah-Jong, potremmo azzardare che i giochi occidentali di carte, sviluppatisi in epoche più recenti, abbiano avuto come antenato proprio questo gioco.

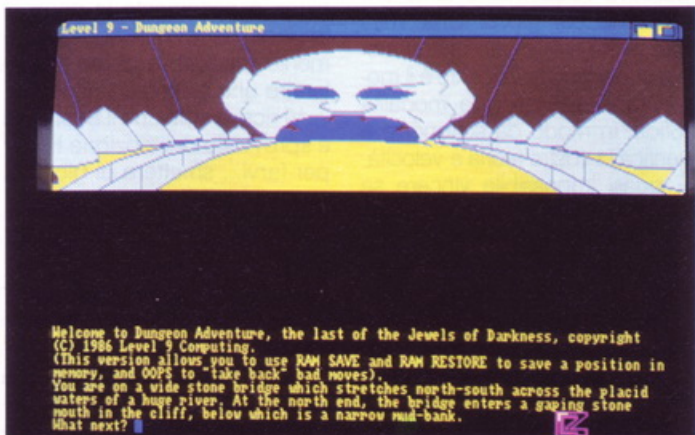
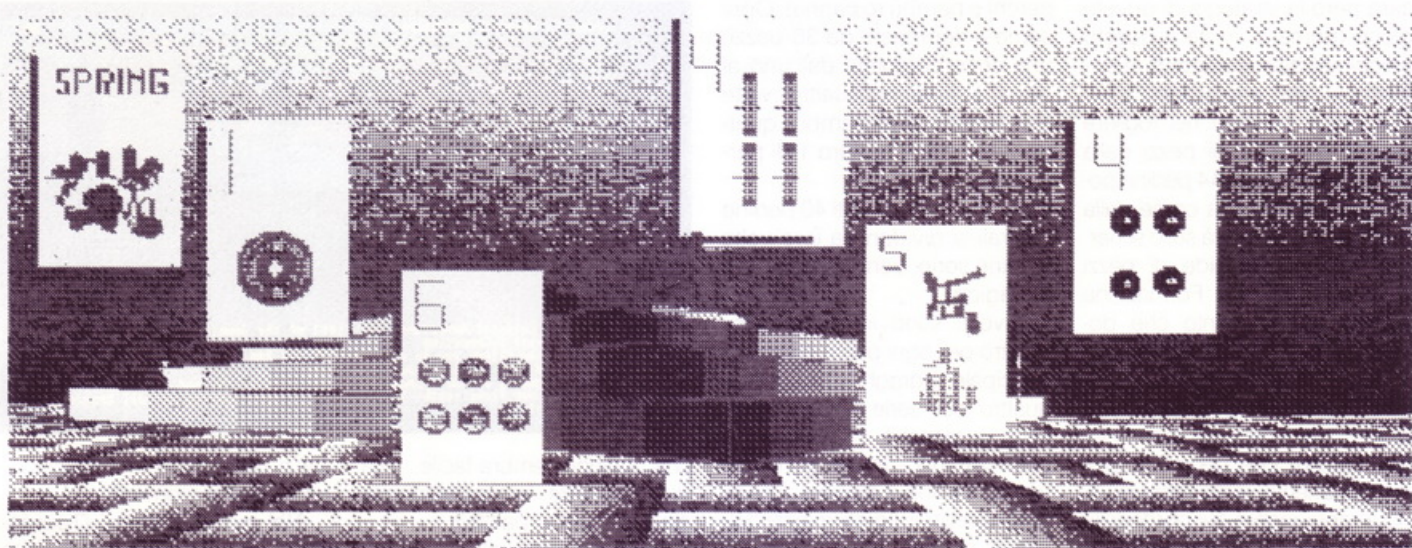
Per vincere non dovrete far altro che accoppiare ciascuna pedina con la sua gemella e rimuovere la coppia dal tavolo; il tutto dovrà essere effettuato in accordo con le regole del gioco e cioè che i pezzi da rimuovere dovranno essere 'liberi' (non dovranno esserci pedine a destra o a sinistra sullo stesso livello del pezzo che si intende rimuovere). Detto così, il gioco sembra facile... ma vi ricrederete immediatamente già dalla prima partita!!!

Il gioco è provvisto di opzioni per aiutarvi a rintracciare le pedine da accopiare, per indicarvi le mosse che vi rimangono, ma se intendete giocare in maniera 'pulita', dovrete rifiutarvi di utilizzare queste opzioni (noi lo abbiamo provato almeno una cinquantina di volte, ma non abbiamo mai vinto!).

Ci sono vari modi per giocare a Shanghai: Solitario, Torneo, Sfida e Team Effort. Ciascuna modalità presenta lo stesso livello di difficoltà, sono variabili solo il numero dei giocatori ed il tempo. 'Solitario' (Solitaire), è la versione per un solo giocatore; in questa modalità non sono presenti le limitazioni di tempo. Il giocatore dovrà solamente togliere dalla configurazione del drago dormiente il maggior numero di pedine possibili; per vincere naturlamente si dovranno togliere tutte le pedine. Nella versione Solitario è possibile registrare una partita e richiamarla in memoria in un secondo momento.

Nel modo 'Torneo' (Tournament), è possibile inserire un numero variabile di giocatori. Ciascun giocatore, toglierà delle pedine da un'identica scacchiera. È possibile scegliere dei tempi limite per torneo di: cinque, dieci, e venti minuti oppure si può scegliere di giocare senza limite di tempo. Il giocatore che è capace di togliere il maggior numero di pezzi vince il torneo. I cinque punteggi migliori saranno registrati automaticamente su disco.

La modalità 'Team Effort' è identica a 'Solitario', unica differenza è che possono collaborare vari giocatori. Ciascun membro del gruppo ha la possibilità di effettuare una mossa, è necessaria perciò la massima collaborazione per raggiungere il successo. La partita terminerà quando uno dei gruppi vince o quando non saranno possibili mosse ulteriori.

'Sfida' (Challenge), è una ga-

ra tra due contendenti per vedere quale dei due effettua il maggior numero di mosse. In questa modalità la strategia del gioco è di vitale importanza dal momento che ciascun giocatore ha un tempo limitato per efftuare la sua mossa. I limiti di tempo per ciascun turno possono essere di 60, 30, 20, o 10 secondi. Il gioco finisce quando ciascun giocatore non toglie alcuna pedina dal gioco per due turni consecutivi.

Personalmente ho giocato con la versione 'Solitario' molte volte in più che con le altre versioni, ciò è dovuto principalmente al fatto che non c'era nessun malcapitato nei dintorni quando giocavo a Shanghai. Il gioco risulta stimolante e normalmente giocavo tre o quattro partite alla volta che mi impegnavano per un paio d'ore circa. Ma il tempo passava velocemente tanto ero assorbito dal gioco. Una volta che avete sviluppato una strategia che reputate vincente è il momento di passare alla modalità 'Sfida' in modo da sottoporre a verifica la vostra teoria e velocità. È quasi impossibile vincere se settate il tempo per effettuare una mossa a dieci secondi, ma dopo un po' questo fatto aumenterà di molto la vostra capacità attentiva e giocando nelle altre modalità otterrete risultati più che confortanti.

Di grande competitività risultano le modalità 'Sfida e Team Effort', questo grazie anche alle limitazioni relative al tempo che vi si possono inserire. Ricordatevi però che nel modo 'Sfida' non si può ricorrere alle opzioni di aiuto se ci si trova nei pasticci. Se invece, pensate di essere dei discreti giocatori, 'Team Effort' sarà il vostro palcoscenico nelle serate con gli amici.

Abbiamo rilevato solamente un piccolo neo in questo programma. Quando non ci sono più mosse a disposizione, il fatto non viene segnalato in alcun modo. Comunque premendo l'opzione HELP si scopre se rimangono ulteriori mosse. Si rischia di guardare lo schermo per una decina di minuti alla ricerca di ulteriori mosse a disposizione, mentre queste non esistono. Se da un lato questo fatto è frustrante, dall'altro offre un ulteriore grado di realisticità al programma. La chiave del successo in questo gioco è certamente la concentrazione, senza questa è praticamente impossibile ideare e mantenere una strategia vincente.

Vi ricordiamo che il Shanghai è sprovvisto dell'opzione HELP per farvi... smettere di giocare.

Silicon Dreams e Jewels of Darkness

Ovvero un pizzico di nostalgia.

Molto tempo fa — prima che la IBM realizzasse che i computer potessero sedersi in cattedra e che i mouse potessero apprendere le arti del disegno, nel tempo in cui si poteva esprimere la memoria di un computer con un numero di due cifre e il numero dei colori veniva espresso con una cifra soltanto — apparve il primo adventure fornito di testo e grafica.

Giochi del tipo Wizard and the Princess e Blade of Black Poole, di compagnie tipo Adventure International e Sierra On-Line, utilizzavano degli analizzatori sintattici molto semplici, e la grafica veniva realizzata con immagini statiche, normalmente sui due terzi superiori dello schermo. Per gli standard a cui siamo abituati, la grafica e l'analisi sintattica di quei giochi sono molto al di sotto della norma. Ma questo accadeva molto tempo fa e come disse una volta qualcuno, "L'unico fatto costante nell'industria dei computer è il cambiamento."

Ora la Firebird Software (Rainbird in Inghilterra) ha ripreso i testi e la grafica classica degli adventure, introducendo due nuovi programmi per l'Amiga: Silicon Dreams e Jewels of Darkness. Il primo è una trilogia d'avventura di Science Fiction (fantascienza) che è ambientata su un mondo chiamato Eden, mentre Jewels of Darkness è una trilogia di avventure fantastiche del tipo cappa e spada ambientate su un mondo chiamato Valaii. Le avventure non sono strettamente collegate tra loro, si può così giocarne una alla volta e in qualsiasi ordine considerandole come giochi separati. Sono comunque più piacevoli se giocate nell'ordine in cui sono presentate dagli autori. C'è la possibilità di ottenere il titolo di 'Supremo Avventurierò se riportate il vostro punteggio da un gioco a quello seguente.

Il gioco ricalca le avventure di Kim Kimberly. Nella prima parte, "Snowball", voi (nelle vesti di Kim) siete stati risvegliati anzitempo dal vostro stato di ibernazione. Siete a bordo della Snowball 9, un'astronave di coloni inviata verso un pianeta chiamato Eden. Qualcuno sta sabotando la navicella, e il computer di bordo pensa che l'unico in grado di intervenire con successo siate voi. Se riuscirete nell'intento, avrete vinto. Poi si passa alla seconda parte, "Return to Eden", inizia con la ciurma della Snowball 9 che si è appena destata e vi sta accusando di tradimento; un nastro danneggiato nella sala di controllo sembra dimostrare che avete cercato di far esplodere l'astronave.

L'episodio finale della trilogia è "Worm in Paradise", per attivare quest'ultima parte dovrete essere stati molto abili in quella precedente e comunque ci vorrà un po' di tempo per raggiungerla. Ora siete diventati

una specie di leggenda, la vostra figura ha assunto proporzioni mitiche.

Jewels of Darkness inizia con "Colossal Adventure, il classico testo d'adventure. Dovrete trovare ed esplorare le Colossal Caverns, poi dovrete immagazzinare il tesoro che avrete trovato nelle caverne in una piccola capanna situata esattamente nelle vicinanze della partenza del gioco. Terminata questa prima parte vi ritroverete in "Adventure Quest", qui dovrete trovare e distruggere il cattivo Lord Demon Agaliarept. La terza parte del gioco, "Dungeon Adventure", inizia immediatamente dopo che avrete assolto al compito precedente, e dovrete esplorare i domini di Lord Demon alla ricerca del tesoro, di magia e di avventura.

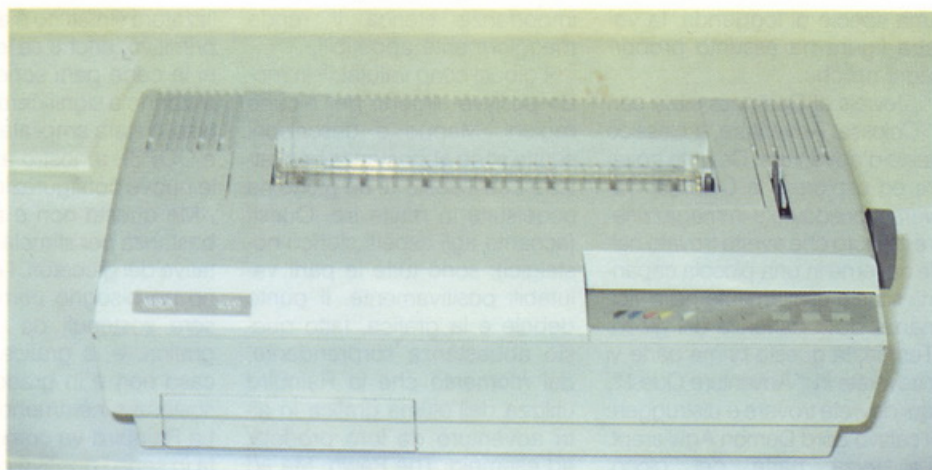
La parte grafica di Silicon Dreams non mi è familiare, ma il primo tracciato di Jewels of Darkness lo è certamente, dal momento che è una variante dell'adventure originale. Questo programma (l'originale, non la versione Rainbird) fa parte del software di pubblico dominio (naturalmente privo di disegni). Sono stati aggiunti alcuni comandi in modo da migliorare quest'ultima versione. Indubbiamente ci sono molte persone che non giocano con computer game da molto tempo e che perciò non hanno mai visto prima questi classici. Sono dei giochi divertenti e la loro importanza storica li rende maggiormente appetibili.

I giochi sono valutabili in modo positivo almeno per alcune ragioni. Mappe e descrizioni scritte sono abbastanza ben fatte, e con il costo di un gioco ne acquistate in realtà tre. Questi (accanto agli aspetti storico-nostalgici), sono tutte le parti valutabili positivamente. Il punto debole è la grafica, fatto questo abbastanza sorprendente, dal momento che la Rainbird utilizza dell'ottima grafica in altri adventure da loro prodotti, ad esempio: The Pawn. Ma ad un'analisi più accurata ci si accorge che The Pawn è stato creato da un altro gruppo di persone, e forse questo fatto può spiegare la differenza.

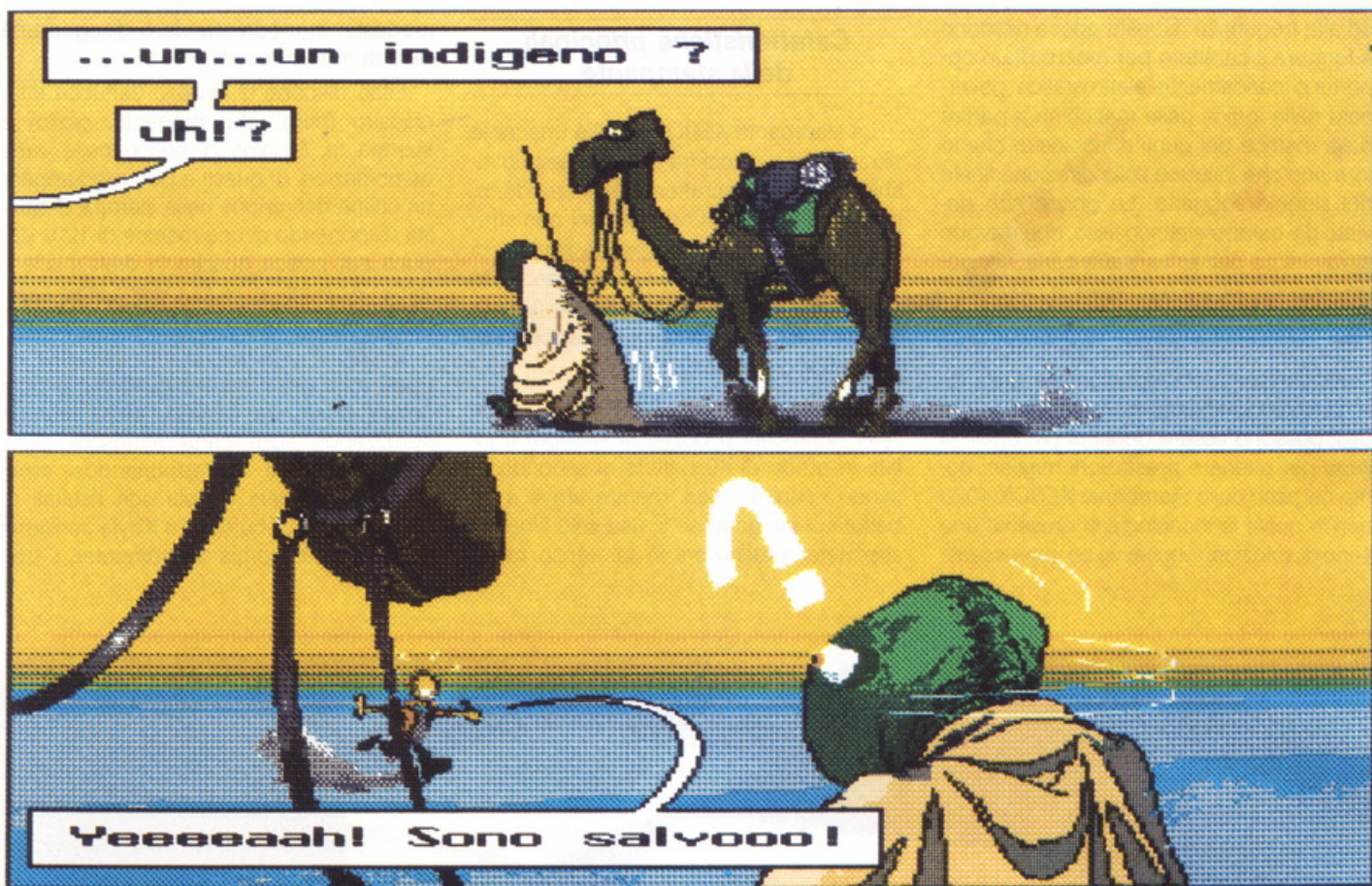
Comunque, i giochi sono accattivanti e le storie interessanti. Al giorno d'oggi i possessori di computer, ed in modo speciale coloro che hanno un Amiga, sono abituati ad una grafica che utilizza un minimo di otto colori (in questo caso ne attendevamo almeno il doppio). Siamo oramai talmente abituati a vedere dei giochi che utilizzino un'ottima grafica che questi giochi rischiano di deludere più di una persona. C'è stato un tempo in cui questi giochi erano il meglio che il mondo dei computer potessero offrire!

Il sistema vi permette di spegnere le figure per mezzo di una selezione da menù. L'analizzatore sintattico è abbastanza primitivo, anche se le descrizioni in certe parti sono gradevoli e stanno a significare che la sintassi è stata ampliata (Ramsave e OOPS!, a nostro avviso sono le nuove configurazioni più utili) . Ma questo non è ancora abbastanza per stimolare le aspettative dei giocatori. I giochi hanno un bisogno primario di essere sostenuti da una ottima grafica, e la grafica in questo caso non è in grado a tratti di sostenere nemmeno se stessa. La Rainbird va comunque citata in senso positivo per un fatto: sul retro della confezione (la quale ci appare come un contenitore per videocassette) è raffigurato un disegno tratto dal gioco. Il disegno non è stato modificato in alcun modo e appare sulla confezione come appare sullo schermo. La software house non ha in alcun modo cercato di celare o di fornire una falsa impressione del gioco. Dal momento che il mercato spesso non corre sui binari dell'onestà, dobbiamo dare credito di ciò alla Rainbird.

E dobbiamo sottolineare anche l'utilità delle nuove configurazioni di Ramsave e OOPS! È utile poter registrare velocemente nella RAM la vostra attuale posizione di gioco, vi libera anche dalla cpia dello schema di protezione. Il tuttovene attivato quando si cerca di effettuare il restore da un disk-based registrato. Questo fatto può risultare noioso quando si effettuano varie operazioni di restore e Ramsave risulta molto utile per questo tipo di situazioni.

OOPS! è ugualmente utile perché vi permette di ritornare alla mossa precedente. In questo modo potete esplorare molto facilmente i luoghi più difficili. È senza dubbio un'ottima configurazione e spero che anche altre compagnie cerchino di implementarla nei testi dei loro adventure. Ci sono vari punti di valore in questi giochi. I giocatori che non sono interessati alla grafica, o che trovino queste storie interessanti, apprezzeranno questi giochi a basso prezzo. Certamente i giocatori più smaliziati non saranno sorpresi dalle modalità dei giochi e dalle strutture di programmazione utilizzate in questa occasione; tali giochi possono comunque essere considerati come un ottimo terreno di allenamento per coloro che hanno appena iniziato la loro carriera con gli adventure. Un'ultima cosa mi preme sottolineare e cioè di non aspettarsi della grafica simile a quella che si può ritrovare in The Pawn, aspettatevi invece di essere riportati ai tempi in cui l'industria dei computer era giovane, e quattro colori erano veramente il massimo.

XEROX 4020

Per avere un delle vostre

di Massimo Lavarian

a stampa più nitida immagini grafiche

Per coloro che sentono la necessità di sviluppare graficamente i loro elaborati, e per quelli che esprimono meglio il loro lavoro con più colori, ad esempio per stampa su fogli di acetato (i lucidi utilizzati poi per proiezioni su lavagne luminose), parleremo di quelle stampanti che meglio soddisfano tali esigenze. Nella famiglia delle stampanti, infatti, considereremo quelle a getto d' inchiostro ed in particolare la XEROX 4020, un prodotto di qualità i cui risultati sono veramente ottimi, anche se il prezzo la fa ritenere un frutto ancora proibito, pur essendo il rapporto prezzo/qualità uno dei migliori per questo tipo di stampanti.

Stampanti a getto d'inchiostro

La scrittura con tale tipo di macchina viene realizzata mediante un flusso di piccole gocce che viene proiettato contro carta di tipo comune, meglio se rispondente ai consigli della Casa Costruttrice. Nel caso del getto continuo, l'inchiostro viene inviato a pressione costante in un serbatoio con un ugello. Per interrompere il flusso di uscita e generare un insieme di gocce equidistanti, viene utilizzato un cristallo piezoelettrico (nel caso della Xerox 4020, il cristallo piezolelettrico viene regolato da un circuito elettronico controllato da un microprocessore) che realizza vibrazioni

ad alta frequenza. Queste gocce riescono a formare il carattere per mezzo di un opportuno caricamento elettrostatico governato dalla logica della macchina a partire dalla matrice dei punti d'inchiostro che lo formano che influisce sulla direzione finale del singolo soggetto. Le gocce non deviate dal campo elettrico fisso, che devono attraversare per arrivare alla carta, vengono raccolte, filtrate e rimesse in circolazione.

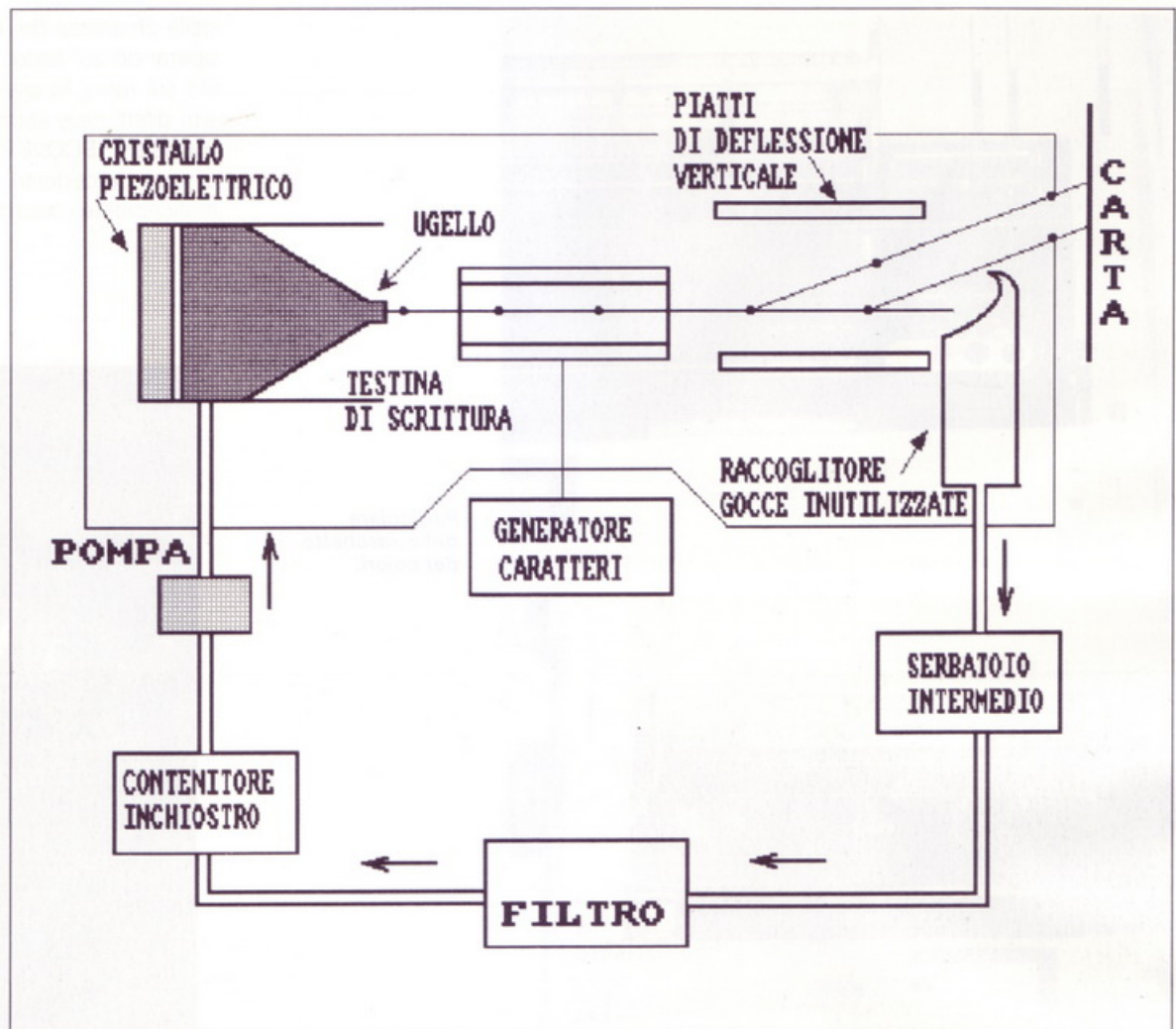
La scrittura del carattere viene eseguita verticalmente e con spostamenti della testina lungo la linea di stampa. Alcuni modelli possono essere dotati di più ugelli in linea per ottenere prestazioni migliori, come nel caso della Stampante XEROX 4020 con la quale la tecnologia ci presenta una serie di soluzioni innovative ed interessanti.

Caratteristiche principali della stampante

La stampante XEROX 4020 è una tra le più versatili e veloci fra quelle a getto d'inchiostro. Le sue caratteristiche la rendono, però, più adatta ad applicazioni con strumenti grafici, o per complesse stampe di elaborati CAD o CAM e via via crescendo per l'image processing ai vari livelli (vedasi tabellina accanto),che non per applicazioni di pura e semplice stampa.

Questo modello, in Draft mode (emissioni di tabulati o bozze di stampa) è infatti in grado di 40 battute al secondo usando i suoi 4 colori, mentre arriva a 80 battute al secondo se si usa solo l'inchiostro nero, prestazioni in tal senso molto modeste; in modo standard disegna una pagina in due minuti.

Esso dispone di quattro cartucce d'inchiostro (nero, blù, magenta, giallo) e stampa in 7 colori diversi e nelle varie combinazioni di questi e può raggiungere un'ottima definizione nella stampa di qualità disponendo di una matrice di 120x120 punti per pollice quadro in draft mode e 240x120 punti in NLQ (near Letter Quality).

Dotata di una testina che presenta 20 ugelli (otto per il nero e quattro per ogni colore base), la XEROX 4020 stampa spruzzando sulla carta minuscoli punti d'inchiostro che, sovrapponendosi, creano caratteri pieni e netti con risultati di buona qualità. Il buffer di 8 Kbyte consente una certa autonomia dal computer, e per-

Schema di funzionamento.

mette di sincronizzare meglio le diverse velocità di produzione e stampa dei dati. Dispone di un insieme completo di 96 caratteri standard ASCII.

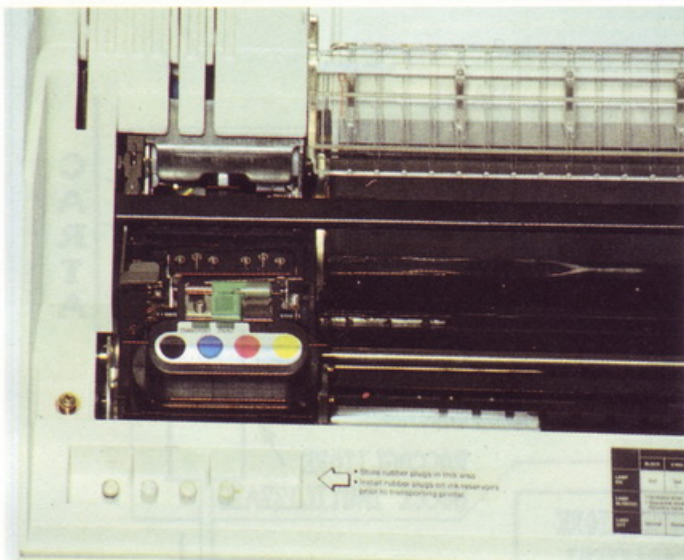
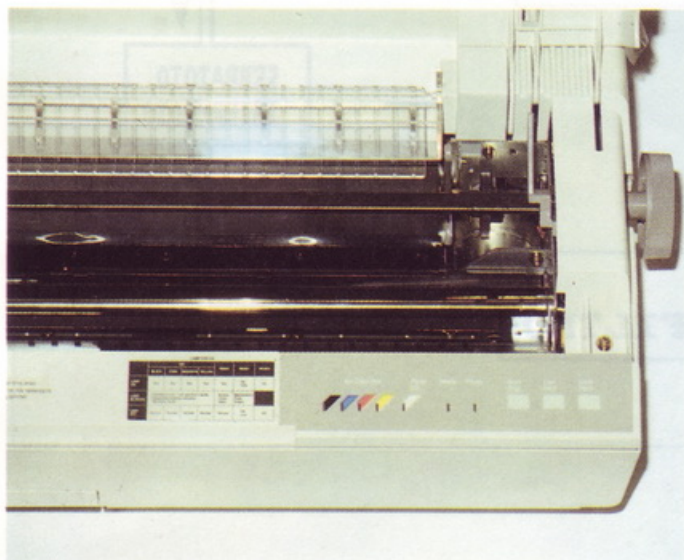
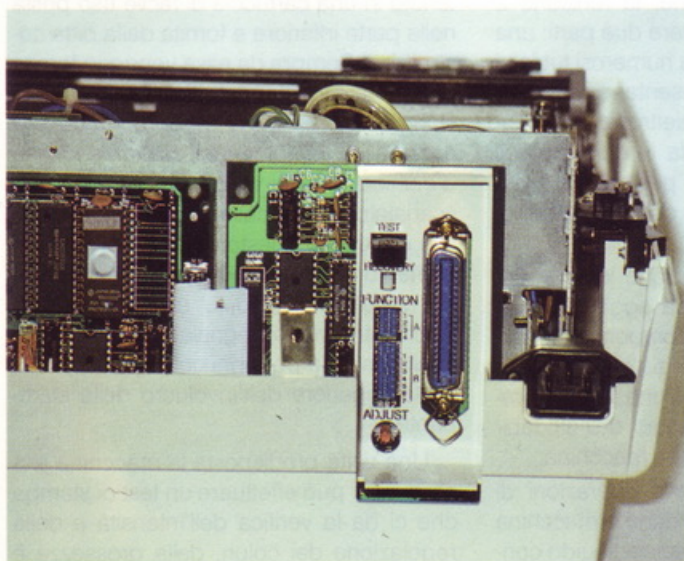
Il pannello di controllo, ridotto all'essenziale, si trova sul lato anteriore, alla destra dell'operatore e consente poche ma fondamentali operazioni. Oltre alla lucetta di accensione e di "pronto stampa" e dei soliti tasti di comando per l'avanzamento della carta, presenta quattro led, ognuno riservato a segnalare l'esaurimento di un colore, ed uno per l' eventuale assenza di carta.

Componentistica

Tolto l'involucro, l'interno della macchina ci rivela un mondo disposto in maniera molto ben ordinata di cui ora vedremo, per sommi capi, la posizione, la funzione e l'uso. Possiamo distinguere due parti: una meccanica composta da numerosi tubicini e da pompe che rappresentano il braccio della macchina ed una elettronica, la mente, posta su una scheda in cui spicca il processore... e trovano posto pure il driver dei motori e la ROM del generatore di caratteri.

Circa la sua composizione interna riteniamo non ci sia altro da aggiungere se non che tutte queste componenti siano disposte in maniera essenziale e razionale, canone fondamentale di una buona progettazione a tutto vantaggio dell'affidabilità e del costo finale della macchina.

La maggior parte delle operazioni di manutenzione che riguardano la macchina sono lasciate ad uno speciale liquido contenuto in una cartuccia di facile uso posta nella parte inferiore e fornita dalla ditta costruttrice. Sempre da essa vengono fornite le cartucce contenenti l'inchiostro per la stampa. Simpatico e molto comodo è il metodo di caricamento delle vaschette d'inchiostro: ogni vaschetta presenta una scanalatura ad incastro per la sua specifica cartuccia, questo onde evitare il caricamento della vaschetta con una cartuccia di colore diverso. L' esaurimento dell'inchiostro viene segnalato da una spia luminosa (una per ogni colore) posta sulla parte superiore dell'involucro della stampante.

Una volta predisposta la macchina alla scrittura si può effettuare un test di stampa che ci da la verifica dell'intensità e della regolazione dei colori, della grossezza e

Particolare delle vaschette dei colori.

Comandi della stampante.

Connettori posti sul retro.

della chiarezza dei caratteri, tutto questo operando sul tasto "TEST SWITCH" posto sul retro. In presenza di sbavature o altri difetti nella stampa si può intervenire sul tasto "RECOVERY SWITCH", posto vicino al precedente. Questo regola automaticamente i colori con un ciclo che dura

Set di carte e colori.

Digitalizzazione in H.A.M. stampata con la Xerox 4020.

circa quattro minuti, compiendo delle procedure di asportazione dell' inchiostro secco dagli ugelli della testina e rimuovendo eventuali presenze d' aria dal sistema. Ad ogni modo la regolazione può essere effettuata pure manualmente intervenendo sugli opportuni Switch.

Ad ogni modo questi interventi si rendono necessari solamente ogni qualvolta si aggiunge l'inchiostro nella vaschetta; infatti la testina, onde evitare problemi legati all'essicamento dell'inchiostro, si posiziona, a stampa ferma, in corrispondenza di un cuscinetto che la mantiene ben chiusa e pronta all'uso dopo essere rimasta inattiva anche per un lungo periodo.

La presenza di un interfaccia parallela Centronics a 8 bit fa si che sia compatibile con i principali sistemi di home computer; è disponibile pure la possibilità d'interfacciamento con la porta seriale RS-232-C.

Il meccanismo di trascinamento della carta è la frizione, il che consente di usarne di qualsiasi tipo nei limiti imposti dalla larghezza del carrello e che raggiunga al massimo 228 mm per riga; è possibile anche optare per l' alimentatore per fogli singoli o per il trattore. Perché non si verifichi l' incoveniente dell'assorbimento del colore da parte della carta con conseguen-

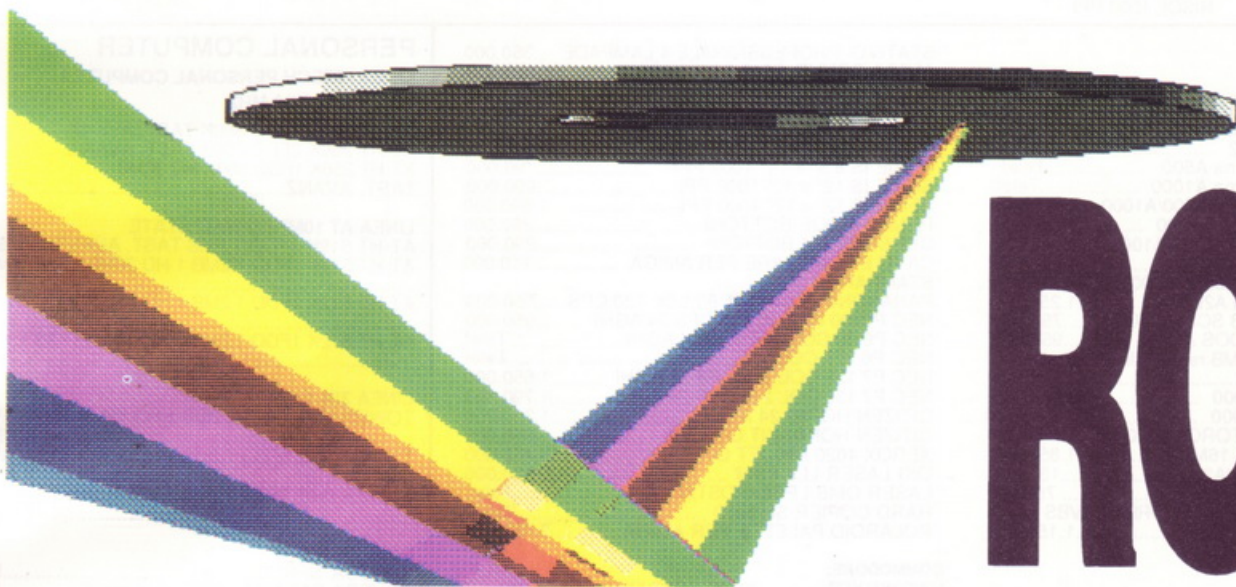
ti possibili sbavature e soprattutto una notevole diminuzione dell'intensità del colore è consigliabile l'uso di una carta speciale distribuita dalla stessa Xerox.

Conclusioni

La stampante XEROX 4020 per le caratteristiche sopra riportate si è dimostrata un ottimo strumento producendo documenti di qualità. Affiancandola ad un Personal Computer AMIGA 2000 si è dimostrata all' altezza della situazione dando il dovuto risalto alle produzioni grafiche di quest'ultimo ed a disegni di qualsiasi tipo. Oltre alle caratteristiche sopracitate non sono state trascurate altre migliorie quali ad esempio la silenziosità (il rumore è inferiore a 55 Db) per cui non impone particolari limiti nella collocazione.

La stampante Xerox 4020 ci è stata gentilmente prestata dalla "Informatica Italia", Corso Re Umberto 128 Torino tel. 501647, che ne gestisce la distribuzione.

Amiga 2000 visto dalla Xerox 4020.

LE ORIGINI DI UN FAMOSO PROGRAMMA:

ROT

Siamo lieti di presentarvi il capostipite dei programmi dedicati alla grafica tridimensionale per Amiga, esso nasce dalla strepitosa mente di Colin French, autore anche delle versioni successive create per il package Aegis Draw. Senz'altro è interessante notare come questo Editor sia nato ancora una volta in un linguaggio tanto criticato, ma poplarissimo come il Basic. Se vi state muovendo nella grafica tridimensionale del vostro Amiga provate questo "programmino" Basic. Con ROT, questo è il nome del programma, potete creare un oggetto composto da 95 punti e da 95 poligoni colorati. Tutto quello che vi necessita sono almeno 512K ed un mouse.

Questo programma richiede molta memoria per cui se dal vostro Workbench non vi risultano almeno 380.000 byte liberi non provate nemmeno a farlo girare. Ricordate che il file "GRAPHICS.BMAP" che si trova nel disco dell'AmigaBasic deve stare sullo stesso dischetto di ROT, spicciatevi dunque a trasferire questo file.

Diamo una rapida occhiata al programma

In ROT l'oggetto è composto da poligoni, come le facce di un cubo. Il poligono viene generato selezionando i punti usati per i suoi vertici. Ogni poligono deve avere almeno tre vertici ma non più di sei. Ogni poligono può avere un differente colore, ed una accurata scelta delle tonalità può produrre dei interessanti effetti d'ombra. Dopo che è stato creato un'oggetto viene disegnata una serie di riquadri (frame) ciascuno contenente l'oggetto in differenti posizioni. Questi riquadri sono catturati come immagini bitmap. Se noi diamo una rapida sequenza à questa serie di immagini otteremo un'azione animata. In ogni frame dovete specificare la rotazione e lo spostamento dell'oggetto lungo le tre assi X, Y e Z. Ad esempio incrementando la rotazione di un frame attorno all'asse Y otterrete un effetto trottola all'esecuzione dell'animazione.

L'Editing dell'Oggetto

Quando Rot viene lanciato per la prima volta compare lo schermo Editor di oggetti, da qui potete passare all'Editor Azione clickando il primo termine del menu Azione. Per ritornare indietro portarsi sul primo termine del menu Oggetto. Alla sinistra dello schermo Editor ci sono tre sezioni dedicate alla posizione dell'oggetto rispetto alle tre assi. Visto dall'alto, di lato e di fronte. Per vedere come questi si incontrano tra loro immaginate di proiettare la figura frontale e quella dall'alto finché i loro angoli non si toccano. Ora avete un mezzo cubo che circonda il vostro oggetto. Ogni rotazione applicata all'oggetto avverrà intorno al centro di questo cubo. Nei tre riquadri della parte sinistra dello schermo c'è visualizzato un cerchio che mette in evidenza il punto selezionato al momento. Molte volte due punti appaiono uno sopra l'altro, controllate che la posizione del punto sia quella realmente da voi voluta.

Vediamo insieme la versione Basic di un programma del package Aegis Draw

di Alessandro Prandi

Un punto con tutte tre le cordinanate settate a zero viene considerato non esistente e quindi non viene visualizzato. Clickando in uno dei tre riquadri si cambieranno due delle tre cordinate del punto. La modifica delle due cordinate dipende dal riquadro in cui si setta il punto. gli angoli del corrente poligono selezionato vengono evidenziati con il colore arancio. Ovviamente se non è stato scelto alcun vertice, il poligono non esiste e pertanto non verrà evidenziato niente.

Nella parte in alto a destra dello schermo di editing dell'oggetto troviamo una serie di controlli per la gestione dei punti. In alto c'è la selezione del punto da modificare, premete il tasto del mouse quando siete posizionati sulle relative frecce e per portarvi al Punto seguente o precedente. Se volete uno spostamento più rapido posizionatevi direttamente nella barra tra le due frecce e clickate con il tasto sinistro. Per settare rapidamente tutte le cordinate al punto zero optate per la casella 'PUNTO ZERO'. In questo modo il punto diventa inesistente, fate quindi attenzione che nessun vertice dei vostri poligoni non usi questo valore come vertice.

Piu' in basso troviamo un altro set di controllo per l'edit dei poligoni. Per creare un poligono bisogna selezionare i punti che ne formano i vertici. Usate l'opzione per la scelta del punto, questo verrà evidenziato nei vari riquadri, quindi premete su '+ il Pt prec.' per gestire il punto come vertice. Un poligono puo' avere dai tre ai sei vertici, gli angoli vengono evidenziati nei vari riquadri. Ricordate che è molto importante la sequenza nella quale selezionate i vari punti. Andate in una direzione lungo il perimetro del poligono, se vedete che gli angoli si incrociano allora saprete che i punti non sono in ordine. Selezionate 'CLR ULTIMO PT'per ripercorrere la lista dei vertici finché non avrete eliminato il problema. Per togliere l'intero poligono, non i punti, optate per 'CLR POLIGONO'.

Nella parte inferiore destra dello schermo c'e' il Palette dei colori. Il colore con il quale verrà disegnato il poligono è evidenziato da un rettangolo arrancione. Per cambiare colore vi sarà sufficiente posizionarvi sul colore scelto e clickarlo. Passiamo ora alle selezioni possibili da menu. Nel menu oggetto ci sono quattro voci, EDITOR OGGETTO, CARICA OGGETTO, SALVA OGGETTO, NUOVO OGGETTO. Lasciamo per ora la prima, e quindi noterete che ci sono le possibilità di salvare una figura editata con ROT e quindi ricaricarla oppure si può cancellare l'oggetto al momento disegnato e ricominciare da capo. Ricordate che gli 'oggetti' vengono salvati con il suffisso .ROTOBJ appeso alla fine del nome del file. Quando caricate i file oggetto non servirà scrivere .ROTOBJ alla fine del nome. Se dimenticate il nome di un file potete scegliere la voce 'FILES' nel menu Rot e cercare di individuarlo mentre la lista scorre sullo schermo.

L'Editing dell'Azione

Selezionate il primo termine del menu Azione per passare allo schermo di edi-

ting dell'azione. La parte superiore e' riservata alla rappresentazione delle figure mentre in basso troviamo una serie di parametri di controllo. A sinistra c'è la selezione del frame, posizionandosi sopra le relative frecce si passerà da un frame all'altro e scegliere con quale lavorare. In ogni frame si possono modificare le rotazioni dell'oggetto rispetto alle tre assi. Per cambiare un valore in particolare basterà clickare sopra lo '0' relativo ed immetere il numero desiderato. I valori immessi vengono controllati dal programma per assicurarsi che essi siano compresi in valori accettabili. Dopo aver inserito i valori una freccia arrancione vi indica la scritta 'Rifà Frame' per comunicarvi che il frame deve essere ridisegnato.

Clickando in 'Rifa' Framè si ordina al programma di ricalcolare l'immagine nel frame in modo che corrisponda ai valori da voi indicati. A questo punto potete passare al frame successivo e ripetere l'operazione, e continuate sinché tutti i 12 frame sono stati ridisegnati. Se ritornate allo schermo dell'Oggetto e ridisegnate un'altra figura e quindi la riportate nello schermo dell'Azione essa non comparirà nei vari frame. Per aggiornarli dovrete selezionare ciascun frame e quindi clickare su 'Rifà frame'; per evitare questa noiosa procedura potete selezionare 'Rifà tutto' ed il programma penserà a completare l'intera sequenza. Dopo aver disegnato tutti e dodici i frame premete il tasto del mouse su 'Play' per vedere l'animazione. Regolate la velocita' di esecuzione agendo sull'indicatore 'Tempo'. Per rivedere la sequenza animata all'infinito selezionate 'Ripete ciclo' nel menu azione, un segno vicino alla scritta comparirà quando questa funzione sarà attivata.

Un altra opzione del menu Azione è quella di 'Ripete ciclo inverso'. Questa voce sta ad indicare che l'animazione partirà dal frame 1 e continuerà fino al 12 e poi ritornerà indietro di nuovo all'1. Questi ultimi due comandi possono essere usati anche contemporaneamente. Per fermare una sequenza animata selezionate 'Stop'.

Gli altri termini contenuti nel menu Azione servono a salvare l'azione dell'oggetto, a caricarla da disco ed a cancellare l'azione corrente. Il file che viene salvato contiene solo i fattori necessari per disegnare il frame. Quando caricate un'azione selezionate 'Rifà tutto' per rigenerare i frame. Essi non vengono salvati perché occuperebero all'incirca 109K. L'operazione richiederebbe più di sei minuti per il caricamento o il save tramite l'AmigaBasic. Speriamo che i comandi BLOAD e BSAVE arrivino presto anche nel Basic Amiga.

L'ultima voce del menu Azione si chiama 'Calcolo tra...' ed ha un ruolo molto importante. Essa calcola e disegna i frame che si trovano tra due figure da voi scelte. Ad esempio se stabilite le coordinate e l'incremento di un oggetto nel frame 1 e nel frame 12 e quindi scegliete questa opzione il programma calcolera' per voi tutti i frame che vanno dal 1 al 12. Settate la rotazione su Y del frame 1 a 90 gradi e quella del frame 7 a 180. Poiché desiderate avere sei figure intermedie che partano da 90 e finiscano con 180 gradi ogni figura ruoterà di 15 gradi rispetto all'asse Y. Naturalmente potete ottenere lo stesso risultato disegnando frame per frame manualmente mentre con 'Calcolo tra...' tutta l'operazione viene condotta automaticamente. Ricordate che il programma sceglie sempre la direzione della rotazione in modo da muovere l'oggetto attraverso l'angolo più piccolo possibile. Se settate il frame iniziale a zero gradi e quello finale a 270 gradi, l'oggetto ruotera di -90 gradi e non di +270.

Diffetti del programma

Alla prima esecuzione delle routine AmigaBasic del programma avrete l'impressione che il computer abbia fatto tilt, visto il tempo che passa dal run all'esecuzione vera e propria. Portate solo un po' di pazienza prima di spegnere definitivamente la macchina. Infatti la seconda volta che una routine viene adoperata i tempi diventano quelli normali. Ci sono alcune parti di ROT, come la routine di caricamento dei file, che sono così lente da farvi sospettare addirittura il peggio. Quando eseguite un'animazione con attivato il ciclo inverso può succedere che ci sia qualche sfarfallamento dell'immagine. Questo diffetto può essere causato dal basic che disegna le immagini con la bitmap nel mezzo della scansione dello schermo. Talvolta cambiare i colori di un oggetto aiuta a minimizzare questo inconveniente.

Gli incrementi di X e Y non sono delle vere e proprie trasformazioni tridimensionali ma solo degli indicatori che servono al disegno del frame sullo schermo. Questa riduzione è necessaria per ridurre le misure delle immagini sullo schermo e principalmente per limitare la memoria richiesta per depositarle. Durante l'animazione il secondo frame si sovrappone al primo, il terzo al secondo e così via, in tal modo viene effettuata la cancellazione dei frame precedenti. Se usate un incremento troppo grande lungo l'asse X di un oggetto piuttosto largo allora noterete che parti di esso non vengono cancellate perfettamente. A questo punto o costruite un'oggetto più piccolo o diminuite l'incremento in X.

Il programma così come lo trovate occupa tutti i 25K di Basic disponibili e molti commenti sono stati tolti dal listato per non andare oltre i limiti consentiti. Purtroppo nen c'e nemmeno lo spazio per un controllo di eventuali errori durante l'I/O del disco, per cui dovete conoscere perfettamente il nome del file da caricare, per una verifica potete usare l'opzione 'Files' nel menu ROT.

Conclusioni

Sicuramente le versioni successive di questo programma, in C, offrono maggiori opportunità di quelle qui esposte però a nostro modesto parere ci pare molto interesante osservare la struttura di questo programma per capire come anche il Basic possa offrire delle bune prestazioni a prescindere dal fattore cronometrico.

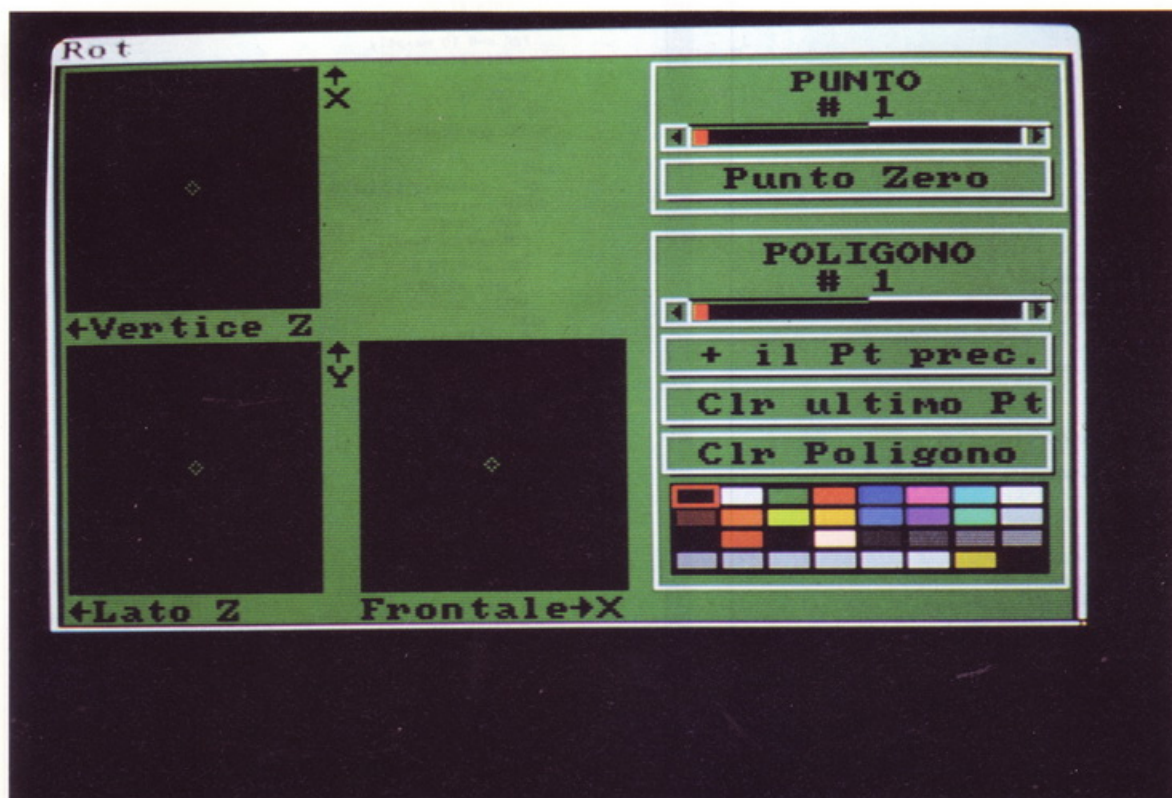
```
GOTO main:
intro:
PRINT "                    R O T                    "
PRINT "                                              "
PRINT "----------------------------------------------"
PRINT "                                              "
PRINT "     Note: le routine in  AmigaBasic"
PRINT "     sono molto lente la prima volta"
PRINT "     che vengono lette."
```

```
PRINT "     Siate pazienti se ROT vi sembra"
PRINT "     troppo stressante."
RETURN
main:
  IF FRE(0)<100000G THEN CLEAR,150000G
  GOSUB init
  quit=0
  WHILE NOT(quit)
    b=MOUSE(0)
```

```
      x=MOUSE(1)
      y=MOUSE(2)
      IF b<>0 THEN
        IF objscr THEN GOSUB edobj
        IF actscr THEN GOSUB edact
      END IF
      m=MENU(0)
      i=MENU(1)
      IF m<>0 THEN GOSUB menuchk
      z$=INKEY$
      IF z$<>"" THEN GOSUB keychk
    WEND
    GOSUB cleanup
END

menuchk:
  ON m GOSUB rotmenu, objmenu, actmenu
RETURN
rotmenu:
  IF i=1 THEN GOSUB listfiles
  IF i=3 THEN quit=(-1)
  RETURN

objmenu:
  IF i=1 AND objscr=0 THEN
    objscr=(-1)
    actscr=0
    MENU 2,1,2
    MENU 2,2,1
    MENU 2,3,1
    MENU 2,4,1
    MENU 3,1,1
    MENU 3,2,0
    MENU 3,3,0
    MENU 3,4,0
    MENU 3,6,0
    MENU 3,7,0
    MENU 3,8,0
    GOSUB drw.objscr
  END IF
  IF i=2 THEN GOSUB loadobj
  IF i=3 THEN GOSUB saveobj
  IF i=4 THEN GOSUB newobj
RETURN
actmenu:
  IF i=1 AND actscr=0 THEN
    actscr=(-1)
    objscr=0
    MENU 3,1,2
    MENU 3,2,1
    MENU 3,3,1
    MENU 3,4,1
    MENU 3,6,1+ABS(actrpt)
    MENU 3,7,1+ABS(actrev)
    MENU 3,8,1
    MENU 2,1,1
    MENU 2,2,0
    MENU 2,3,0
    MENU 2,4,0
    FOR n=1 TO 12
      frmchg(n)=1
    NEXT
    GOSUB drw.actscr
  END IF
  IF i=2 THEN GOSUB loadact
  IF i=3 THEN GOSUB saveact
  IF i=4 THEN GOSUB newact
  IF i=6 THEN
    actrpt=NOT(actrpt)
    MENU 3,6,1+ABS(actrpt)
  END IF
  IF i=7 THEN
    actrev=NOT(actrev)
    MENU 3,7,1+ABS(actrev)
  END IF
  IF i=8 THEN GOSUB calctween
RETURN
listfiles:
  s$="File in:"
  GOSUB drw.filereq
  s$="DF0:"
  GOSUB getstring2
  IF s$<>"c" AND s$<>"C" AND s$<>"" THEN
    CLS
    FILES s$
    PRINT
    GOSUB click.continue
    IF objscr<>0 THEN
      GOSUB drw.objscr
    ELSE
      GOSUB drw.actscr
    END IF
    GOSUB nobut
```

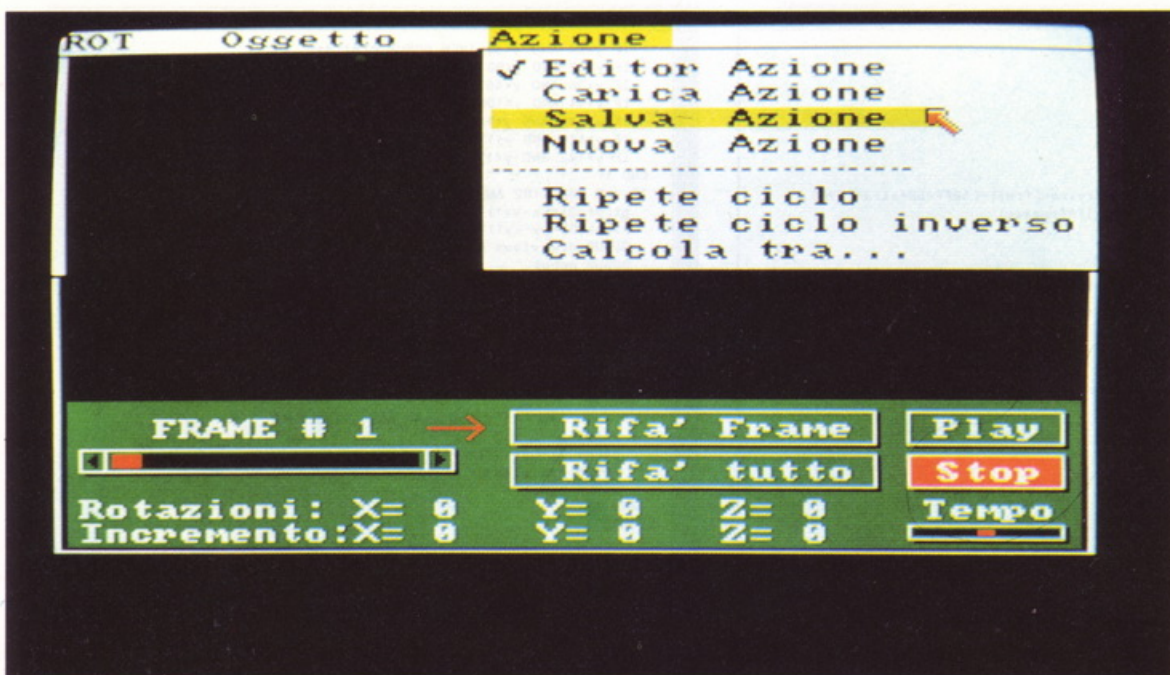
```
    END IF
  RETURN
  loadobj:
    s$="Load:"
    GOSUB drw.filereq
    GOSUB getstring
    IF s$<>"c" AND s$<>"C" AND s$<>"" THEN
      s$=s$+".ROTOBJ"
      OPEN s$ FOR INPUT AS #1
      FOR n=0 TO 95
        FOR n2=0 TO 3
          INPUT#1,pt(n,n2)
        NEXT
      NEXT
      FOR n=0 TO 95
        FOR n2=0 TO 6
          INPUT#1,poly(n,n2)
        NEXT
      NEXT
      FOR n=0 TO 95
        INPUT#1,polyclr(n)
      NEXT
      FOR n=0 TO 95
        INPUT#1,vrt(n)
      NEXT
      CLOSE #1
      pt=1
      poly=1
    END IF
    GOSUB drw.objscr
  RETURN
  saveobj:
    s$="Save:"
    GOSUB drw.filereq
    GOSUB getstring
    IF s$<>"c" AND s$<>"C" AND s$<>"" THEN
      s$=s$+".ROTOBJ"
      OPEN s$ FOR OUTPUT AS #1
      FOR n=0 TO 95
        FOR n2=0 TO 3
          PRINT#1,pt(n,n2);
        NEXT
      NEXT
      FOR n=0 TO maxpoly
        FOR n2=0 TO 6
          PRINT#1,poly(n,n2);
        NEXT
      NEXT
      FOR n=0 TO maxpoly
        PRINT#1,polyclr(n);
      NEXT
      FOR n=0 TO maxpoly
        PRINT#1,vrt(n);
      NEXT
      CLOSE #1
    END IF
    GOSUB drw.objscr
  RETURN
  newobj:
    s$="   Cancello l'oggetto?"
    GOSUB you.sure
    IF sure THEN
      FOR n=0 TO maxpt
        FOR n2=0 TO 2
          pt(n,n2)=0
        NEXT
      NEXT
      FOR n=0 TO maxpoly
        FOR n2=0 TO 6
          poly(n,n2)=0
        NEXT
        polyclr(n)=0
        vrt(n)=0
      NEXT
      pt=1
      poly=1
    END IF
    GOSUB drw.objscr
  RETURN

loadact:
  s$="Load:"
  GOSUB drw.filereq
  GOSUB getstring
  IF s$<>"c" AND s$<>"C" AND s$<>"" THEN
    s$=s$+".ROTACT"
    OPEN s$ FOR INPUT AS #1
    FOR n=1 TO 12
      INPUT#1,xrot(n),yrot(n),zrot(n)
      INPUT#1,xtran(n),ytran(n),ztran(n)
    NEXT
    INPUT#1,spd,actrpt,actrev
    CLOSE#1
```

```
      frm=1
      FOR n=1 TO 12
        frmchg(n)=1
      NEXT
      GOSUB drw.frmnum
      GOSUB drw.spdnum
      GOSUB drw.factors
      GOSUB drw.update
      MENU 3,6,1+ABS(actrpt)
      MENU 3,7,1+ABS(actrev)
    END IF
    LINE (0,0)-(311,131),0,bf
    GOSUB putfrm
  RETURN

  saveact:
    s$="Save:"
    GOSUB drw.filereq
    GOSUB getstring
    IF s$<>"c" AND s$<>"C" AND s$<>"" THEN
      s$=s$+".ROTACT"
      OPEN s$ FOR OUTPUT AS #1
      FOR n=1 TO 12
        PRINT#1,xrot(n);yrot(n);zrot(n);
        PRINT#1,xtran(n);ytran(n);ztran(n);
      NEXT
      PRINT#1,spd;actrpt;actrev;
      CLOSE #1
    END IF
    LINE(0,0)-(311,131),0,bf
    GOSUB putfrm
  RETURN

  newact:
    s$="   Cancello l'azione?"
    GOSUB you.sure
    IF sure THEN
      FOR n=0 TO 12
        xrot(n)=0:yrot(n)=0:zrot(n)=0
        xtran(n)=0:ytran(n)=0:ztran(n)=0
      NEXT
      spd=20
```

```
      actrpt=0
      actrev=0
      MENU 3,6,1
      MENU 3,7,1
      frm=1
    END IF
    GOSUB drw.actscr
  RETURN

  calctween:
    GOSUB gettween
    IF stfrm>endfrm THEN SWAP stfrm,endfrm
    stp=endfrm-stfrm
    LINE(0,0)-(311,131),0,bf
    IF stp>1 THEN
      xrot=xrot(endfrm)-xrot(stfrm)
      IF xrot>180 THEN xrot=(360-xrot)*(-1)
      IF xrot<-180 THEN xrot=xrot+360
      stpxrot=xrot/stp
      yrot=yrot(endfrm)-yrot(stfrm)
      IF yrot>180 THEN yrot=(360-yrot)*(-1)
      IF yrot<-180 THEN yrot=yrot+360
      stpyrot=yrot/stp
      zrot=zrot(endfrm)-zrot(stfrm)
      IF zrot>180 THEN zrot=(360-zrot)*(-1)
      IF zrot<-180 THEN zrot=zrot+360
      stpzrot=zrot/stp
      xtran=xtran(endfrm)-xtran(stfrm)
      stpxtran=xtran/stp
      ytran=ytran(endfrm)-ytran(stfrm)
      stpytran=ytran/stp
      ztran=ztran(endfrm)-ztran(stfrm)
      stpztran=ztran/stp
      FOR frm=stfrm+1 TO endfrm-1
        n=frm-stfrm
        xrot(frm)=xrot(stfrm)+INT(stpxrot*n)
        IF xrot(frm)>359 THEN xrot(frm)=xrot(frm)-360
        IF xrot(frm)<0 THEN xrot(frm)=xrot(frm)+360
        yrot(frm)=yrot(stfrm)+INT(stpyrot*n)
        IF yrot(frm)>359 THEN yrot(frm)=yrot(frm)-360
        IF yrot(frm)<0 THEN yrot(frm)=yrot(frm)+360
        zrot(frm)=zrot(stfrm)+INT(stpzrot*n)
```

Finestra oggetto.

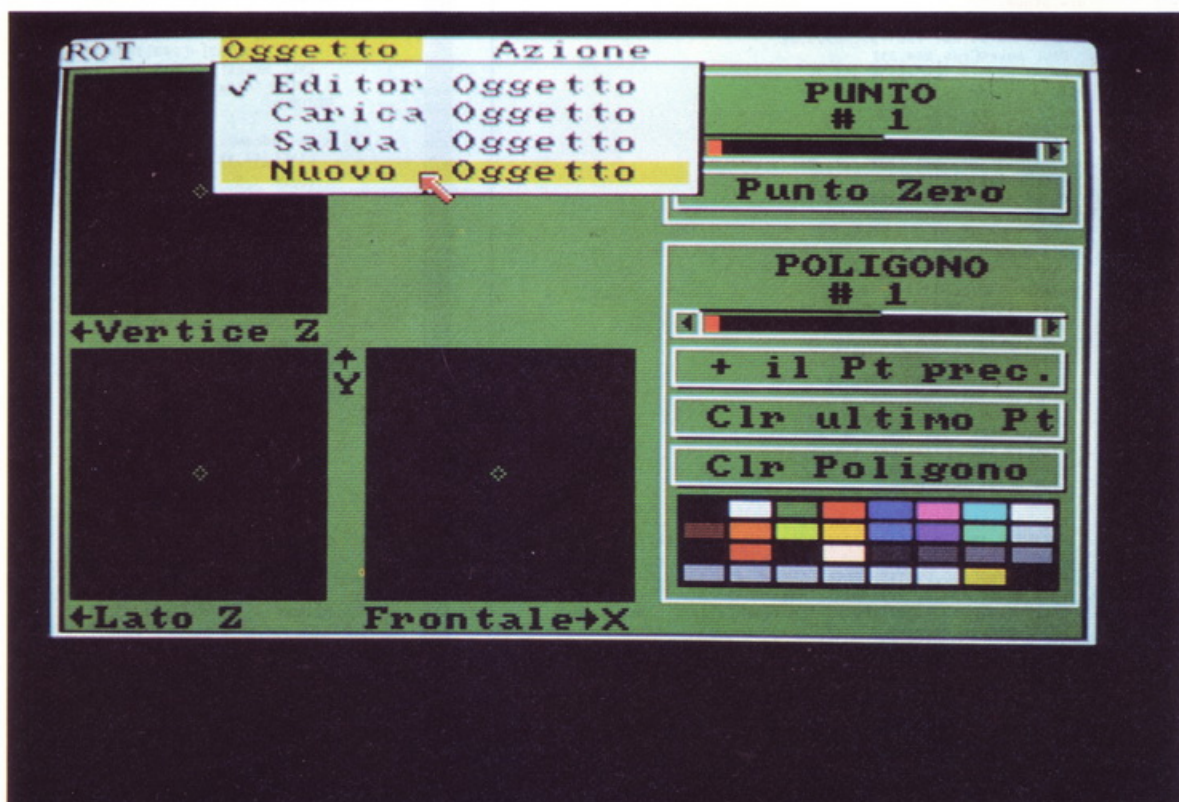
```
        IF zrot(frm)>359 THEN zrot(frm)=zrot(frm)-360
        IF zrot(frm)<0 THEN zrot(frm)=zrot(frm)+360
        xtran(frm)=xtran(stfrm)+INT(stpxtran*n)
        ytran(frm)=ytran(stfrm)+INT(stpytran*n)
        ztran(frm)=ztran(stfrm)+INT(stpztran*n)
        GOSUB drw.frmnum
        GOSUB drw.factors
        GOSUB high.redraw
        GOSUB drw.frame
        GOSUB getfrm
        frmchg(frm)=0
        GOSUB drw.update
        GOSUB unhigh.redraw
      NEXT
      frm=stfrm
      GOSUB drw.frmnum
      GOSUB drw.factors
    END IF
    LINE(0,0)-(311,131),0,bf
    GOSUB putfrm
  RETURN

  gettween:
    GOSUB drw.tweenreq
    maxchar=2
    xt=188:yt=74
    GOSUB getstring
    stfrm=VAL(s$)
    IF stfrm<1 THEN stfrm=1
    IF stfrm>12 THEN stfrm=12
    yt=82
    GOSUB getstring
    endfrm=VAL(s$)
    IF endfrm<1 THEN endfrm=1
    IF endfrm>12 THEN endfrm=12
  RETURN

  drw.tweenreq:
    LINE(58,48)-(254,92),0,bf
    LINE(60,50)-(252,90),3,bf
    LINE(61,51)-(251,89),2,bf
    CALL move&(rp&,76,66)
    PRINT "Calcolo intermedio"
    CALL move&(rp&,92,74)
    PRINT " Dal Frame:"
    CALL move&(rp&,108,82)
    PRINT "Al Frame:"
  RETURN

  drw.filereq:
    LINE(50,48)-(262,92),0,bf
    LINE(52,50)-(260,90),3,bf
    LINE(53,51)-(259,89),2,bf
    CALL move&(rp&,60,66)
    PRINT "      File Requestor"
      IF s$<>"File in:" THEN
        CALL move&(rp&,92,74)
        PRINT "('C' per uscire)"
      END IF
    CALL move&(rp&,60,82)
    PRINT s$
    xt=68+LEN(s$)*8
    yt=82
    maxchar=23-LEN(s$)
  RETURN
    frmchg(frm)=0
    GOSUB drw.update
  NEXT
  frm=tfrm
  GOSUB drw.frmnum
  GOSUB drw.factors
  GOSUB putfrm
  GOSUB unhigh.redraw2
  GOSUB nobut
RETURN

playbut:
  GOSUB high.play
  GOSUB unhigh.stop
  GOSUB freeze.menu
  frminc=1
  clickstop=0
  WHILE NOT(clickstop)
    frm=frm+frminc
    IF frm>12 THEN
      IF actrev THEN
        frm=11
        frminc=(-1)
      ELSEIF actrpt THEN
        frm=1
      ELSE
        frm=1
        clickstop=(-1)
      END IF
      END IF
      IF frm<1 THEN
      IF actrpt THEN
        frm=2
        frminc=1
      ELSE
        frm=1
        clickstop=(-1)
      END IF
    END IF
    GOSUB putfrm
    GOSUB drw.frmnum
    FOR n=0 TO 39-spd
      b=MOUSE(0)
      x=MOUSE(1)
      y=MOUSE(2)
      IF b<>0 THEN
        IF x>260 AND x<306 AND y>152 AND y<164 THEN
          clickstop=(-1)
          n=39-spd
        END IF
        IF x>263 AND x<303 AND y>178 AND y<183 THEN
          spd=x-263
          GOSUB drw.spdnum
        END IF
      END IF
    NEXT
  WEND
  GOSUB drw.factors
  GOSUB drw.update
  GOSUB unhigh.play
  xt=108:yt=184
  maxchar=4
  numonly=1
  GOSUB getstring2
  GOSUB high.stop
  GOSUB unfreeze.menu
  GOSUB nobut
RETURN

stopbut:
RETURN

spdslider:
  spd=x-263
  GOSUB drw.spdnum
RETURN

mod.xrot:
  s$=STR$(xrot(frm))
  xt=108:yt=175
  maxchar=4
  numonly=1
  GOSUB getstring2
  xrot(frm)=VAL(s$)
  xrot(frm)=xrot(frm) MOD 360
  IF xrot(frm)<0 THEN xrot(frm)=xrot(frm)+360
  GOSUB drw.factors
  frmchg(frm)=1
  GOSUB drw.update
  GOSUB nobut
RETURN

mod.yrot:
  s$=STR$(yrot(frm))
  xt=164:yt=175
  maxchar=4
  numonly=1
  GOSUB getstring2
  yrot(frm)=VAL(s$)
  yrot(frm)=yrot(frm) MOD 360
  IF yrot(frm)<0 THEN yrot(frm)=yrot(frm)+360
  GOSUB drw.factors
  frmchg(frm)=1
  GOSUB drw.update
  GOSUB nobut
RETURN

mod.zrot:
  s$=STR$(zrot(frm))
  xt=220:yt=175
  maxchar=4
  numonly=1
  GOSUB getstring2
  zrot(frm)=VAL(s$)
  zrot(frm)=zrot(frm) MOD 360
  IF zrot(frm)<0 THEN zrot(frm)=zrot(frm)+360
  GOSUB drw.factors
  frmchg(frm)=1
  GOSUB drw.update
  GOSUB nobut
RETURN

mod.xtran:
  s$=STR$(xtran(frm))
```

```
    xtran(frm)=VAL(s$)
    IF xtran(frm)<-90 THEN xtran(frm)=(-90)
    IF xtran(frm)>90 THEN xtran(frm)=90
    GOSUB drw.factors
    frmchg(frm)=1
    GOSUB drw.update
    GOSUB nobut
RETURN

mod.ytran:
    s$=STR$(ytran(frm))
    xt=164:yt=184
    maxchar=3
    numonly=1
    GOSUB getstring2
    ytran(frm)=VAL(s$)
    IF ytran(frm)<-8 THEN ytran(frm)=(-8)
    IF ytran(frm)>8 THEN ytran(frm)=8
    GOSUB drw.factors
    frmchg(frm)=1
    GOSUB drw.update
    GOSUB nobut
RETURN

mod.ztran:
    s$=STR$(ztran(frm))
    xt=220:yt=184
    maxchar=4
    numonly=1
    GOSUB getstring2
    ztran(frm)=VAL(s$)
    IF ztran(frm)<0 THEN ztran(frm)=0
    IF ztran(frm)>999 THEN ztran(frm)=999
    GOSUB drw.factors
    frmchg(frm)=1
    GOSUB drw.update
    GOSUB nobut
RETURN

drw.frame:
    GOSUB unit.matrix
    IF xrot(frm)>0 THEN GOSUB apply.xrot
    IF yrot(frm)>0 THEN GOSUB apply.yrot
    IF zrot(frm)>0 THEN GOSUB apply.zrot
    IF ztran(frm)>0 THEN GOSUB apply.ztran
    GOSUB transform.pts
    GOSUB convert2scr
    GOSUB sort.poly
    LINE(0,0)-(311,131),0,bf
    GOSUB drw.object
RETURN

matprep:
   FOR row=0 TO 3
      FOR col=0 TO 3
         tr1(row,col)=tran(row,col)

         tr2(row,col)=0
      NEXT
   NEXT
RETURN

unit.matrix:
   FOR row=0 TO 3
      FOR col=0 TO 3
         tran(row,col)=0
         IF row=col THEN tran(row,col)=1
      NEXT
   NEXT
RETURN

matmult:
   FOR row=0 TO 3
      FOR col=0 TO 3
         t=0
         FOR e1=0 TO 3
            t=t+tr1(row,e1)*tr2(e1,col)
         NEXT
         tran(row,col)=t
      NEXT
   NEXT
RETURN

apply.xrot:     'ruota l'oggetto intorno all'asse X
   GOSUB matprep
   rad=xrot(frm)*3.1416/180
   tr2(0,0)=1:tr2(3,3)=1
   tr2(1,1)=COS(rad):tr2(1,2)=SIN(rad)*(-1)
   tr2(2,1)=SIN(rad):tr2(2,2)=COS(rad)
   GOSUB matmult
RETURN

apply.yrot:     'ruota l'oggetto intorno all'asse Y
   GOSUB matprep
   rad=yrot(frm)*3.1416/180
   tr2(1,1)=1:tr2(3,3)=1
   tr2(0,0)=COS(rad):tr2(0,2)=SIN(rad)
   tr2(2,0)=SIN(rad)*(-1):tr2(2,2)=COS(rad)
   GOSUB matmult
RETURN

apply.zrot:     'ruota l'oggetto intorno all'asse Z
   GOSUB matprep
   rad=zrot(frm)*3.1416/180
   tr2(2,2)=1:tr2(3,3)=1
   tr2(0,0)=COS(rad):tr2(0,1)=SIN(rad)*(-1)
   tr2(1,0)=SIN(rad):tr2(1,1)=COS(rad)
   GOSUB matmult
RETURN

apply.ztran:     'trasporta l'oggetto lungo l'asse Z
   GOSUB matprep
   tr2(0,0)=1:tr2(1,1)=1
   tr2(2,2)=1:tr2(3,3)=1
   tr2(3,2)=ztran(frm)
   GOSUB matmult
RETURN
transform.pts:
  COLOR 1,0
  FOR p=1 TO 95
    LOCATE 1,5
    PRINT "Calcola il Punto";p
    IF pt(p,0)<>0 OR pt(p,1)<>0 OR pt(p,2)<>0 THEN
      FOR col=0 TO 3
        t=0
        FOR e1=0 TO 3
          t=t+pt(p,e1)*tran(e1,col)
        NEXT
        tpt(p,col)=t
      NEXT
    END IF
  NEXT
RETURN

convert2scr:     ' converte i punti sulle coordinate dello schermo
  FOR p=1 TO 95
    r=zeye/(tpt(p,2)+zeye)
    tpt(p,0)=INT(tpt(p,0)*r)+hoff
    tpt(p,1)=(INT(tpt(p,1)*r))*(-1)+voff
  NEXT
RETURN

reset.polyorder:
  FOR n=1 TO maxpoly
    polyord(n,0)=0
    polyord(n,1)=0
  NEXT
RETURN

sort.poly:
  GOSUB reset.polyorder
  FOR n=1 TO maxpoly
    LOCATE 1,16
    PRINT "Poligono"n
    IF vrt(n)>0 THEN
      t=0
      FOR n2=1 TO vrt(n)
        t=t+tpt(poly(n,n2),2)
      NEXT
      polyord(n,0)=INT(t/vrt(n))
    END IF
  NEXT
  FOR n=1 TO maxpoly
    LOCATE 1,24
    PRINT n;"di nuovo."
    IF vrt(n)>0 THEN
      t=(-100)
      p=(-1)
      FOR n2=1 TO maxpoly
        IF vrt(n2)>0 THEN
          IF polyord(n2,0)>t THEN t=polyord(n2,0):p=n2
        END IF
      NEXT
      polyord(n,1)=p
      polyord(p,0)=(-100)
    END IF

  NEXT
  COLOR 1,2
RETURN

drw.object:
  FOR n=1 TO maxpoly
    IF vrt(n)>2 THEN
      FOR n2=1 TO vrt(polyord(n,1))
        AREA(tpt(poly(polyord(n,1),n2),0)+xtran(frm),tpt(poly(polyord
(n,1),n2),1)+ ytran(frm))
      NEXT
```

```
      COLOR polyclr(polyord(n,1))
      AREAFILL
    END IF
  NEXT
  COLOR 1,2
RETURN

getfrm:
  GET(hoff-64+xtran(frm),voff-58+ytran(frm))-(hoff+63+xtran(frm),
voff+57+ytran(frm)),frame&((frm-1)*frmsize)
RETURN

putfrm:
  PUT(hoff-64+xtran(frm),voff-58+ytran(frm)),frame&((frm-1)*frmsize)
,PSET
RETURN

getstring:
  s$=""

getstring2:
  GOSUB freeze.menu
  GOSUB nokey
  numchar=LEN(s$)
  CALL move&(rp&,xt,yt)
  PRINT s$;
  z$=""

getstring3:
    LINE(xt+numchar*8+1,yt-7)-(xt+numchar*8+4,yt+2),3,bf
    z$=INPUT$(1)
    LINE(xt+numchar*8+1,yt-7)-(xt+numchar*8+4,yt+2),2,bf
    IF z$=CHR$(8) OR z$=CHR$(31) OR z$=CHR$(127) THEN
      IF numchar>0 THEN
        PRINT CHR$(8);" ";CHR$(8);
        numchar=numchar-1
        s$=LEFT$(s$,numchar)
      END IF
    END IF
    IF ASC(z$)>31 AND numchar<maxchar AND numonly=0 THEN
        s$=s$+z$
        PRINT z$;
        numchar=numchar+1
    END IF
    IF ASC(z$)>31 AND numchar<maxchar THEN
      IF numonly=1 THEN
        IF (z$>="0" AND z$<="9") OR z$="-" THEN
          s$=s$+z$
          PRINT z$;
    you.sure:
      GOSUB drw.surereq
      GOSUB nobut
      answer=0
      WHILE NOT(answer)
        b=MOUSE(0)
        x=MOUSE(1)
        y=MOUSE(2)
        IF b<>0 THEN
          IF y>80 AND y<92 THEN
            IF x>75 AND x<127 THEN
              sure=(-1)
              answer=(-1)
            END IF
            IF x>187 AND x<239 THEN
              sure=0
              answer=(-1)
            END IF
          END IF
          GOSUB nobut
        END IF
      WEND
    RETURN

    drw.surereq:
      LINE(43,48)-(270,100),0,bf
      LINE(45,50)-(268,98),3,bf
      LINE(46,51)-(267,97),2,bf
      CALL move&(rp&,53,66)
      PRINT "      Siete sicuri!!!"
      CALL move&(rp&,53,74)
      PRINT LEFT$(s$,24)
      LINE(75,80)-(127,92),1,b
      LINE(187,80)-(239,92),1,b
      CALL move&(rp&,93,89)
      PRINT "SI"
      CALL move&(rp&,190,89)
      PRINT "  NO"
    RETURN

    keychk:
      RETURN

    edobj:
      IF x>189 AND x<306 THEN
        IF y>24 AND y<32 THEN GOSUB ptslider
        IF y>82 AND y<90 THEN GOSUB polyslider
        IF y>36 AND y<48 THEN GOSUB zeropt
        IF y>94 AND y<106 THEN GOSUB addpt
        IF y>110 AND y<122 THEN GOSUB undopt
        IF y>126 AND y<138 THEN GOSUB delpoly
        IF y>142 AND y<171 THEN GOSUB selclr
      END IF
      IF x>3 AND x<82 AND y>3 AND y<82 THEN
        pt(pt,2)=(x-vx1)*(-1)
        pt(pt,0)=(y-vy1)*(-1)
        GOSUB drw.views
        GOSUB nobut
      END IF
      IF x>3 AND x<82 AND y>96 AND y<175 THEN

          pt(pt,2)=(x-vx2)*(-1)
          pt(pt,1)=(y-vy2)*(-1)
          GOSUB drw.views
          GOSUB nobut
        END IF
        IF x>97 AND x<176 AND y>96 AND y<175 THEN
          pt(pt,0)=x-vx3
          pt(pt,1)=(y-vy3)*(-1)
          GOSUB drw.views
          GOSUB nobut
        END IF
      RETURN

      zeropt:
        p=pt
        GOSUB unhigh.pt
        GOSUB erase.pt
        pt(pt,0)=0
        pt(pt,1)=0
        pt(pt,2)=0
        GOSUB drw.pt
        GOSUB high.pt
      RETURN

      addpt:
        IF vrt(poly)>5 THEN BEEP:RETURN
        IF pt(pt,0)=0 AND pt(pt,1)=0 AND pt(pt,2)=0 THEN RETURN
        vrt(poly)=vrt(poly)+1
        poly(poly,vrt(poly))=pt
        GOSUB drw.views
        GOSUB nobut
      RETURN

      undopt:
        IF vrt(poly)>0 THEN
          poly(poly,vrt(poly))=0
          vrt(poly)=vrt(poly)-1
        END IF
        GOSUB drw.views
        GOSUB nobut
      RETURN

      delpoly:
        IF vrt(poly)>0 THEN
          c=polyclr(poly)
          GOSUB unhigh.clr
          FOR n=0 TO vrt(poly)
            poly(poly,n)=0
          NEXT
          vrt(poly)=0
          polyclr(poly)=0
          c=polyclr(poly)
          GOSUB high.clr
        END IF
        GOSUB drw.views
        GOSUB nobut
      RETURN

      selclr:
        IF x>191 AND x<304 THEN

          c=INT((x-192)/14)+INT((y-143)/7)*8
          IF c<>polyclr(poly) THEN
            GOSUB high.clr
            SWAP c,polyclr(poly)
            GOSUB unhigh.clr
          END IF
        END IF
      RETURN

      drw.views:
        GOSUB erase.views
        p=pt
        GOSUB high.pt
        FOR p=1 TO maxpt
          GOSUB drw.pt
        NEXT
        p=pt
        t=poly
```

Finestra azione.

```
    FOR poly=1 TO maxpoly
      GOSUB drw.poly
    NEXT
    poly=t
    GOSUB high.poly
  RETURN

  erase.views:
    LINE(2,2)-(83,83),0,bf
    LINE(2,95)-(83,176),0,bf
    LINE(96,95)-(177,176),0,bf
  RETURN

  drw.pt:
    IF pt(p,0)<>0 OR pt(p,1)<>0 OR pt(p,2)<>0 THEN
      PSET(vx1-pt(p,2),vy1-pt(p,0))
      PSET(vx2-pt(p,2),vy2-pt(p,1))
      PSET(vx3+pt(p,0),vy3-pt(p,1))
    END IF
  RETURN

  erase.pt:
    COLOR 0
    GOSUB drw.pt
    COLOR 1
  RETURN

  high.pt:
    CIRCLE(vx1-pt(p,2),vy1-pt(p,0)),2,2
    CIRCLE(vx2-pt(p,2),vy2-pt(p,1)),2,2
    CIRCLE(vx3+pt(p,0),vy3-pt(p,1)),2,2
  RETURN

  unhigh.pt:
    CIRCLE(vx1-pt(p,2),vy1-pt(p,0)),2,0
    CIRCLE(vx2-pt(p,2),vy2-pt(p,1)),2,0
    CIRCLE(vx3+pt(p,0),vy3-pt(p,1)),2,0
  RETURN

  drw.poly:

   IF vrt(poly)>0 THEN

     PSET(vx1-pt(poly(poly,1),2),vy1-pt(poly(poly,1),0))
     PSET(vx2-pt(poly(poly,1),2),vy2-pt(poly(poly,1),1))
     PSET(vx3+pt(poly(poly,1),0),vy3-pt(poly(poly,1),1))
     IF vrt(poly)>1 THEN
       FOR n=2 TO vrt(poly)
         LINE(vx1-pt(poly(poly,n-1),2),vy1-pt(poly(poly,n-1),0))
-(vx1-pt(poly(poly,n),2),vy1-pt(poly(poly,n),0))
         LINE(vx2-pt(poly(poly,n-1),2),vy2-pt(poly(poly,n-1),1))
-(vx2-pt(poly(poly,n),2),vy2-pt(poly(poly,n),1))
         LINE(vx3+pt(poly(poly,n-1),0),vy3-pt(poly(poly,n-1),1))
-(vx3+pt(poly(poly,n),0),vy3-pt(poly(poly,n),1))
       NEXT
       LINE(vx1-pt(poly(poly,n-1),2),vy1-pt(poly(poly,n-1),0))
-(vx1-pt(poly(poly,1),2),vy1-pt(poly(poly,1),0))
       LINE(vx2-pt(poly(poly,n-1),2),vy2-pt(poly(poly,n-1),1))
-(vx2-pt(poly(poly,1),2),vy2-pt(poly(poly,1),1))
       LINE(vx3+pt(poly(poly,n-1),0),vy3-pt(poly(poly,n-1),1))
-(vx3+pt(poly(poly,1),0),vy3-pt(poly(poly,1),1))
     END IF
   END IF
  RETURN

  erase.poly:
    COLOR 3
    GOSUB drw.poly
    COLOR 1
  RETURN

  high.poly:
    COLOR 3
    GOSUB drw.poly
    RETURN

  unhigh.poly:
    GOSUB drw.poly
  RETURN

  high.c.r:
    y2=INT(c/8)
    x2=c-y2*8
    LINE(192+x2*14,143+y2*7)-(205+x2*14,149+y2*7),3,b
  RETURN

  unhigh.clr:
    y2=INT(c/8)
    x2=c-y2*8
    LINE(192+x2*14,143+y2*7)-(205+x2*14,149+y2*7),0,b
  RETURN

  pislider:
    IF x<197 THEN
      p=pt
      GOSUB unhigh.pt
      pt=pt-1
      IF pt<1 THEN pt=1
      p=pt
      GOSUB drw.ptnum
      GOSUB high.pt
      GOSUB nobut
```

```
        ELSEIF x>298 THEN
          p=pt
          GOSUB unhigh.pt
          pt=pt+1
          IF pt>maxpt THEN pt=maxpt
          p=pt
          GOSUB drw.ptnum
          GOSUB high.pt
          GOSUB nobut
        ELSEIF x>199 AND x<295 THEN
          p=pt
          GOSUB unhigh.pt
          pt=x-199
          p=pt
          GOSUB drw.ptnum
          GOSUB high.pt
          GOSUB nobut
        END IF
      RETURN

      polyslider:
        IF x<197 THEN
          c=polyclr(poly)
          GOSUB unhigh.clr
          GOSUB unhigh.poly
          poly=poly-1
          IF poly<1 THEN poly=1
          GOSUB drw.polynum
          GOSUB high.poly
          c=polyclr(poly)
          GOSUB high.clr
          GOSUB nobut
        ELSEIF x>298 THEN
          c=polyclr(poly)
          GOSUB unhigh.clr
          GOSUB unhigh.poly
          poly=poly+1
          IF poly>maxpoly THEN poly=maxpoly
          GOSUB drw.polynum
          GOSUB high.poly
          c=polyclr(poly)
          GOSUB high.clr
          GOSUB nobut
```

```
        ELSEIF x>199 AND x<295 THEN
          c=polyclr(poly)
          GOSUB unhigh.clr
          GOSUB unhigh.poly
          poly=x-199
          GOSUB drw.polynum
          GOSUB high.poly
          c=polyclr(poly)
          GOSUB high.clr
          GOSUB nobut
        END IF
      RETURN
      edact:
        IF x>4 AND x<122 AND y>150 AND y<158 THEN GOSUB frmslider:RETURN
        IF x>40 AND x<250 AND y>136 AND y<148 THEN GOSUB drw.frm:RETURN
        IF x>140 AND x<250 AND y>152 AND y<164 THEN GOSUB drw.allfrm:RETURN
        IF x>260 AND x<306 AND y>136 AND y<148 THEN GOSUB playbut:RETURN
        IF x>260 AND x<306 AND y>152 AND y<164 THEN GOSUB stopbut:RETURN
        IF x>263 AND x<303 AND y>178 AND y<183 THEN GOSUB spdslider:RETURN
        IF y>168 AND y<176 THEN
          IF x>92 AND x<140 THEN GOSUB mod.xrot:RETURN
          IF x>147 AND x<196 THEN GOSUB mod.yrot:RETURN
          IF x>203 AND x<252 THEN GOSUB mod.zrot:RETURN
        END IF
        IF y>177 AND y<185 THEN
          IF x>92 AND x<140 THEN GOSUB mod.xtran:RETURN
          IF x>147 AND x<196 THEN GOSUB mod.ytran:RETURN
          IF x>203 AND x<252 THEN GOSUB mod.ztran:RETURN
        END IF
      RETURN

      frmslider:
        IF x<12 THEN
          frm=frm-1
          IF frm<1 THEN frm=1
          GOSUB drw.frmnum
          GOSUB drw.update
          GOSUB drw.factors
          GOSUB putfrm
          GOSUB nobut
        ELSEIF x>114 THEN
          frm=frm+1
          IF frm>12 THEN frm=12
```

Finestra oggetto.

```
      GOSUB drw.frmnum
      GOSUB drw.update
      GOSUB drw.factors
      GOSUB putfrm
      GOSUB nobut
    ELSEIF x>16 AND x<110 THEN
      frm=INT((x-16)/8)+1
      GOSUB drw.frmnum
      GOSUB drw.update
      LINE(0,0)-(311,131),0,bf
      GOSUB drw.factors
      GOSUB putfrm
      GOSUB nobut
    END IF
  RETURN

  drw.frm:
    GOSUB high.redraw
    GOSUB drw.frame
    GOSUB getfrm
    frmchg(frm)=0
    GOSUB drw.update
    GOSUB unhigh.redraw
    GOSUB nobut
  RETURN

  drw.allfrm:
    GOSUB high.redraw2
    tfrm=frm
    FOR frm=1 TO 12
      GOSUB drw.frmnum
      GOSUB drw.factors
      GOSUB drw.frame
      GOSUB getfrm
  drw.leftarrow:
    AREA(x,y):AREA(x+3,y-3):AREA(x+3,y+3):AREAFILL
    LINE(x,y)-(x+6,y)
  RETURN

  drw.rightarrow:
    AREA(x,y):AREA(x-3,y-3):AREA(x-3,y+3):AREAFILL
    LINE(x,y)-(x-6,y)
  RETURN

  drw.uparrow:
    AREA(x,y):AREA(x-3,y+3):AREA(x+3,y+3):AREAFILL
    LINE(x,y)-(x,y+6)
  RETURN

  drw.ptnum:
    LINE(199,26)-(296,30),0,bf
    LINE(198+pt,26)-(201+pt,30),3,bf
    CALL move&(rp&,244,21)
    PRINT RIGHT$("00"+STR$(pt),2)
  RETURN

  drw.polynum:
    LINE(199,84)-(296,88),0,bf
    LINE(198+poly,84)-(201+poly,88),3,bf
    CALL move&(rp&,244,79)
    PRINT RIGHT$("00"+STR$(poly),2)
  RETURN

  drw.actscr:
    LINE(0,0)-(311,131),0,bf
    LINE(0,132)-(311,186),2,bf
    x=140:y=136:GOSUB drw.button2
    y=152:GOSUB drw.button2
    x=260:y=136:GOSUB drw.button3
    y=152:GOSUB drw.button3
    LINE(260,178)-(306,183),1,b
    LINE(262,184)-(307,184),0
    LINE -(307,179),0
    LINE(4,150)-(122,158),1,b
    LINE(12,150)-(114,158),1,b
    LINE(6,159)-(123,159),0
    LINE -(123,151),0
    COLOR 0
    AREA(7,154):AREA(9,152):AREA(9,156):AREAFILL
    AREA(117,152):AREA(119,154):AREA(117,156):AREAFILL
    COLOR 1,2
    CALL move&(rp&,28,145):PRINT "FRAME #"
    CALL move&(rp&,4,175):PRINT "Rotazioni: X=       Y=      Z="
    CALL move&(rp&,4,184):PRINT "Incremento:X=       Y=      Z=";
    GOSUB unhigh.redraw
    GOSUB unhigh.redraw2
    GOSUB unhigh.play
    GOSUB high.stop
    CALL move&(rp&,264,175):PRINT "Tempo"
    GOSUB drw.frmnum
    GOSUB drw.spdnum
    GOSUB drw.update
    GOSUB drw.factors
    GOSUB putfrm
```

```
  RETURN

  drw.frmnum:
    LINE(15,152)-(111,156),0,bf
    LINE(7+frm*8,152)-(15+frm*8,156),3,bf
    CALL move&(rp&,84,145)
    PRINT RIGHT$("00"+STR$(frm),2)
  RETURN

  drw.spdnum:
    LINE(262,180)-(304,181),0,b
    LINE(261+spd,180)-(265+spd,181),3,b
  RETURN
  drw.update:
    LINE(115,138)-(131,146),2,bf
    IF frmchg(frm)<>0 THEN
      LINE(115,142)-(131,142),3
      LINE -(127,138),3
      LINE(131,142)-(127,146),3
    END IF
  RETURN

  drw.factors:
    CALL move&(rp&,108,175)
    PRINT LEFT$(STR$(xrot(frm))+"    ",4)
    CALL move&(rp&,164,175)
    PRINT LEFT$(STR$(yrot(frm))+"    ",4)
    CALL move&(rp&,220,175)
    PRINT LEFT$(STR$(zrot(frm))+"    ",4)
    CALL move&(rp&,108,184)
    PRINT LEFT$(STR$(xtran(frm))+"    ",4);
    CALL move&(rp&,164,184)
    PRINT LEFT$(STR$(ytran(frm))+"    ",4);
    CALL move&(rp&,220,184)
    PRINT LEFT$(STR$(ztran(frm))+"    ",4);
  RETURN

  unhigh.redraw:
    LINE(141,137)-(249,147),2,bf
    CALL move&(rp&,148,145)
    PRINT " Rifa' Frame"
  RETURN

  high.redraw:
    LINE(141,137)-(249,147),3,bf
    COLOR 1,3
    CALL move&(rp&,148,145)
    PRINT " Rifa' Frame"
    COLOR 1,2
  RETURN

  unhigh.redraw2:
    LINE(141,153)-(249,163),2,bf
    CALL move&(rp&,156,161)
    PRINT "Rifa' tutto"
  RETURN

  high.redraw2:
    LINE(141,153)-(249,163),3,bf
    COLOR 1,3
    CALL move&(rp&,156,161)
    PRINT "Rifa' tutto"
    COLOR 1,2
  RETURN

  unhigh.play:
    LINE(261,137)-(305,147),2,bf
    CALL move&(rp&,268,145)
    PRINT "Play"
  RETURN

  high.play:
    LINE(261,137)-(305,147),3,bf
    COLOR 1,3
    CALL move&(rp&,268,145)
    PRINT "Play"
    COLOR 1,2
  RETURN

  unhigh.stop:
    LINE(261,153)-(305,163),2,bf
    CALL move&(rp&,268,161)
    PRINT "Stop"
  RETURN

  high.stop:
    LINE(261,153)-(305,163),3,bf
    COLOR 1,3
    CALL move&(rp&,268,161)
    PRINT "Stop"
    COLOR 1,2
  RETURN

  drw.button2:
```

```
    LINE(x,y)-(x+110,y+12),1,b
    LINE(x+2,y+13)-(x+111,y+13),0
    LINE -(x+111,y+1),0
  RETURN

  drw.button3:
    LINE(x,y)-(x+46,y+12),1,b
    LINE(x+2,y+13)-(x+47,y+13),0
    LINE -(x+47,y+1),0
  RETURN
            numchar=numchar+1
          END IF
        END IF
      END IF
    IF z$<>CHR$(13) THEN getstring3
    numonly=0
    GOSUB unfreeze.menu
  RETURN

  click.continue:
    LOCATE 21,4
    PRINT "                Per continuare       "
    PRINT
    PRINT "  premere il tasto sinistro del mouse";
    GOSUB nobut
    b=MOUSE(0)
    WHILE b=0
      b=MOUSE(0)
    WEND
  RETURN

  nobut:
    b=MOUSE(0)
    WHILE b<>0
      b=MOUSE(0)
    WEND
  RETURN

  nokey:
    z$=INKEY$
    WHILE z$<>""
      z$=INKEY$
    WEND
  RETURN
  init:
    DEFINT a-p,u-z
    DECLARE FUNCTION setdrmd LIBRARY
    DECLARE FUNCTION move LIBRARY
    LIBRARY "df1:graphics.library"
    SCREEN 1,320,200,5,1
    WINDOW 2,"Rot",(0,0)-(311,186),0,1
    WINDOW OUTPUT 2
    rp&=WINDOW(8) 'puntatore alla porta raster
    PALETTE 0,0,0,0
    PALETTE 2,0,.5,0
    PALETTE 31,0,.25,0
    PALETTE 30,.7,.7,0
    LOCATE 3,1
    GOSUB intro
    DIM pt(95,3),poly(95,6),polyclr(95),vrt(95)
    DIM xrot(12),yrot(12),zrot(12)
    DIM xtran(12),ytran(12),ztran(12),frmchg(12)
    DIM tran(3,3),tr1(3,3),tr2(3,3),tpt(95,3),polyord(95,1)
    DIM frame&(27876)
    pt=1:poly=1
    objscr=-1:actscr=0
    vx1=43:vx2=43:vx3=136
    vy1=43:vy2=135:vy3=135
    maxpt=95:maxpoly=95
    frm=1:spd=20
    frmsize=2323
    hoff=156:voff=66:zeye=440
    actrpt=0:actrev=0
    FOR n=1 TO maxpt
      pt(n,3)=1
    NEXT
    GOSUB init.menu
    COLOR 1,0
    GOSUB click.continue
    COLOR 1,2
    GOSUB drw.objscr
  RETURN

  init.menu:
    MENU 1,0,1,"ROT"
    MENU 1,1,1," Files "
    MENU 1,2,0,"------"
    MENU 1,3,1," Esci "
    MENU 2,0,1,"Oggetto"
    MENU 2,1,2,"  Editor Oggetto "
    MENU 2,2,1,"  Carica Oggetto "
    MENU 2,3,1,"  Salva  Oggetto "
    MENU 2,4,1,"  Nuovo  Oggetto "
    MENU 3,0,1,"Azione"
    MENU 3,1,1,"  Editor Azione "
    MENU 3,2,0,"  Carica Azione "
    MENU 3,3,0,"  Salva  Azione "
    MENU 3,4,0,"  Nuova  Azione "
    MENU 3,5,0,"-----------------"
    MENU 3,6,0,"  Ripete ciclo "
    MENU 3,7,0,"  Ripete ciclo inverso"
    MENU 3,8,0,"  Calcola tra...   "
    MENU 4,0,0," "
    MENU 4,1,0," "
  RETURN

  freeze.menu:
    MENU 1,0,0
    MENU 2,0,0
    MENU 3,0,0
  RETURN

  unfreeze.menu:
    MENU 1,0,1
    MENU 2,0,1
    MENU 3,0,1
  RETURN

  cleanup:
    WINDOW CLOSE 2
    SCREEN CLOSE 1
    LIBRARY CLOSE
    PALETTE 0,0,.25,.55
    PALETTE 2,0,0,0
    MENU RESET
  RETURN

  drw.objscr:
    LINE(0,0)-(320,200),2,bf
  LINE(2,2)-(83,83),0,bf
  LINE(2,95)-(83,176),0,bf
  LINE(96,95)-(177,176),0,bf
  LINE(186,2)-(310,53),1,b
  LINE(186,60)-(310,176),1,b
  LINE(191,142)-(304,171),0,bf
  FOR y=0 TO 3
    FOR x=0 TO 7
      LINE(193+x*14,144+y*7)-(204+x*14,148+y*7),y*8+x,bf
    NEXT
  NEXT
  x=189:y=126:GOSUB drw.button
  y=110:GOSUB drw.button
  y=94:GOSUB drw.button
  y=36:GOSUB drw.button
  y=24:GOSUB drw.scroll
  y=82:GOSUB drw.scroll
  CALL move&(rp&,228,12):PRINT "PUNTO"
  CALL move&(rp&,236,21):PRINT "#"
  CALL move&(rp&,208,45):PRINT "Punto Zero"
  CALL move&(rp&,220,70):PRINT "POLIGONO"
  CALL move&(rp&,236,79):PRINT "#"
  CALL move&(rp&,200,103):PRINT "+ il Pt prec."
  CALL move&(rp&,200,119):PRINT "Clr ultimo Pt"
  CALL move&(rp&,192,135):PRINT " Clr Poligono"
  CALL move&(rp&,10,92):PRINT "Vertice Z"
  CALL move&(rp&,10,185):PRINT "Lato Z";
  CALL move&(rp&,96,185):PRINT "Frontale X";
  CALL move&(rp&,86,16):PRINT "X"
  CALL move&(rp&,86,109):PRINT "Y"
  x=2:y=89:GOSUB drw.leftarrow
  y=182:GOSUB drw.leftarrow
  x=89:y=2:GOSUB drw.uparrow
  y=95:GOSUB drw.uparrow
  x=166:y=182:GOSUB drw.rightarrow
  GOSUB drw.ptnum
  GOSUB drw.polynum
  GOSUB drw.views
  c=polyclr(poly)
  GOSUB high.clr
  GOSUB nobut
RETURN

drw.button:
  LINE(x,y)-(x+116,y+12),1,b
  LINE(x+2,y+13)-(x+117,y+13),0
  LINE -(x+117,y+1),0
RETURN

drw.scroll:
  LINE(x,y)-(x+116,y+8),1,b
  LINE(x+8,y)-(x+109,y+8),1,b
  LINE -(x+117,y+1),0
  COLOR 0
  AREA(x+3,y+4):AREA(x+5,y+2):AREA(x+5,y+6)
  AREAFILL
  AREA(x+112,y+2):AREA(x+112,y+6):AREA(x+114,y+4)
  AREAFILL
RETURN
```

ADEGUATI AI PROGETTI AMBRA ED ERGON

NICA SUPERIORE

GLI ISTITUTI TECNICI

Paul B. Zbar
Joseph G. Sloop
LABORATORIO DI ELETTRONICA INTEGRATA
NIE 681405 X
pp. 246 • L. 18.000
Le caratteristiche dei circuiti integrati lineari e degli amplificatori operazionali. Vengono proposte esperienze per utilizzare circuiti integrati per la generazione dei segnali e i circuiti PLL. Si mostrano alcune utilizzazioni dei circuiti integrati digitali e dei convertitori A/D e D/A.

Renzo Traversini
MICROELETTRONICA: TECNOLOGIE E DISPOSITIVI
Corso di tecnologie elettroniche Vol. II
NIE 681126 W
pp. 192 • L. 18.000
Le tecnologie più utilizzate per la fabbricazione dei circuiti integrati al silicio, presentate in stretto legame con le strutture fisiche dei componenti (diodi, transistor bipolari, transistori MOS) che con esse si realizzano.

Mariangela Botti
DAL PROBLEMA AL PROGRAMMA
NIE 681352 J
pp. 328 • L. 24.000
Obiettivo è quello di fornire agli allievi che iniziano lo studio dell'informatica gli elementi fondamentali per la risoluzione di un problema sino alle soglie della sua codifica. I numerosi esercizi risolti e proposti giocano un ruolo di assoluto rilievo: attraverso l'analisi e la descrizione degli algoritmi vengono presentate le strutture fondamentali della programmazione, gli array e le subroutine.

Felice Tarantini
COMMUTAZIONE TELEFONICA AUTOMATICA
NIE 681403 Q
pp. 220 • L. 23.000
Fornisce agli studenti la conoscenza delle tecniche di commutazione automatica affermatesi in Italia a partire dai primi sistemi automatici fino agli attuali sistemi interamente elettronici, evidenziando l'aspetto funzionale tecnologico degli autocommutatori. Vengono poi descritti il funzionamento della rete telefonica nazionale e le tecniche di commutazione nelle centrali.

Paul B. Zbar
Joseph G. Sloop
LABORATORIO DI ELETTROTECNICA
NIE 681399 M
pp. 302 • L. 21.000
L'uso del multimetro, la realizzazione di alcune reti elettriche, le caratteristiche di alcuni campi elettrici e magnetici, l'utilizzo dell'oscilloscopio, alcune esercitazioni sui condensatori, sugli induttori e quindi sui circuiti RL, RC, RLC serie e parallelo.

Fosco Bellomo
ELEMENTI PASSIVI TECNOLOGIE E DISPOSITIVI
Corso di tecnologie elettroniche Vol. I
NIE 681457 P
pp. 352 • L. 24.000
Vengono introdotti i fondamenti tecnologici relativi ai materiali utilizzati nel campo elettronico e ai parametri meccanici, fisici e chimici che ne determinano la scelta. Vengono analizzate le tecnologie costruttive degli elementi cosiddetti passivi, quali resistori, condensatori, induttori, legandole alle strutture fisiche dei componenti e agli aspetti applicativi degli stessi.

Peter Bishop
INFORMATICA GENERALE
NIE 681473 J
pp. 540 • L. 24.000
Cod.: SD668
Tutti gli aspetti teorici e pratici della materia informatica, come previsto dai programmi ministeriali per gli Istituti Tecnici Industriali e Commerciali. Si articola in cinque sezioni: i principi dell'elaborazione dell'informazione, la struttura dell'elaboratore e l'architettura dei sistemi, il software di sistema, l'organizzazione dei dati e le applicazioni.

Giuseppe Saccardi
TELEMATICA DAI PROTOCOLLI ALLE RETI
NIE 681449 X
pp. 240 • L. 24.000
Il mondo della telematica partendo dall'evoluzione verso le reti telematiche, descrivendo i protocolli di trasmissione sincroni oggi più usati, cioè i BSC, l'SDLC, l'HDLC. Sono descritti i dispositivi per reti telematiche, di multiplazione e di concentrazione, il che permette di comprendere il passaggio da una rete convenzionale ad una commutazione di pacchetto.

Paul B. Zbar
Joseph G. Sloop
LABORATORIO DI ELETTRONICA DI BASE
NIE 681401 W
pp. 272 • L. 18.000
Il funzionamento dei multimetri e degli oscilloscopi per la misura delle grandezze elettriche fondamentali, esercitazioni sui diodi a semiconduttori sui circuiti elettronici che li utilizzano, esperimenti sugli alimentatori, sui transistori a funzione e ad effetto di campo.

Fosco Bellomo
MICROELETTRONICA NUOVE TECNOLOGIE
Corso di tecnologie elettroniche Vol. III
NIE 681467 W
pp. 200 • L. 18.000
L'obiettivo in questo testo è quello di fornire, a completamento dei programmi ministeriali del triennio degli Istituti Tecnici, una conoscenza approfondita sulle nuove tecnologie, con la descrizione delle mete raggiunte e dei vantaggi ottenuti ed ipotizzando, in alcuni casi, quali potranno essere i successivi sviluppi.

Salvatore Consentino
ORGANIZZAZIONE INDUSTRIALE STUDI DI FABBRICAZIONE E DISEGNO
NIE 681463 K
pp. 216 • L. 22.000
Si articola in tre parti: la prima mette in rilievo l'importanza della grafica e delle tecniche di lavoro legate al CAD. Una seconda parte è dedicata alla struttura dell'impresa industriale nelle sue principali funzioni e sono descritti limiti e vantaggi dei modelli nella ricerca operativa. L'ultima sezione riguarda gli aspetti relativi alla produzione.

Paul H. Young
COMUNICAZIONI ELETTRICHE
Corso di radioelettronica
NIE 681465 M
pp. 498 • L. 34.000
Partendo dalla descrizione degli amplificatori a radiofrequenza, degli oscillatori, degli spettri dei segnali e dei sistemi di modulazione d'ampiezza, si passa alla trattazione dei più moderni circuiti di trasmissione e di ricezione. Ampio spazio viene dedicato alla comunicazione digitale e alle tecniche di trasmissione dati.

umenti
che cambia

DISTRIBUZIONE ESCLUSIVA:
La Nuova Italia

GRUPPO EDITOR

ELETTRONICA E AUTOMAZIONE · IN
TECNOLOGIE E MERCATI ·

Il Gruppo Editoriale Jackson S.p.A. è il primo e più importante editore italiano specializzato nell'area delle nuove tecnologie, entro cui offre un range completo di prodotti e servizi, che garantiscono la risposta precisa a qualsiasi domanda di conoscenza o aggiornamento da parte dell'utente, sia esso neofita o esperto professionista.

Le riviste Jackson si articolano da quest'anno in quattro aree editoriali specifiche strutturate per mercato, al fine di offrire al lettore notizie e servizi sempre più approfonditi e adeguati alla posizione di assoluta leadership del Gruppo Editoriale Jackson, su scala nazionale e internazionale.

Il recente accordo con la multinazionale dell'editoria tecnica V N U Business Press Group assicura, infatti, alle riviste Jackson l'accesso a un network internazionale di notizie e informazioni tecniche in ogni specifico settore di intervento editoriale, indispensabile per parlare di nuove tecnologie offrendo il massimo dell'aggiornamento e della professionalità. Queste le quattro aree editoriali Jackson per il 1989, cui si

RIALE JACKSON

FORMATICA E PERSONAL COMPUTER

HOBBY E HOME COMPUTER

aggiungono, ovviamente, l'area Libri e Grandi Opere e l'area Formazione, che assieme alle riviste, consentono a Jackson di proporsi come l'unico vero "sistema editoriale integrato nell'High Technology":
ELETTRONICA E AUTOMAZIONE • EO News Settimanale • Elettronica Oggi • Automazione Oggi • Strumentazione e Misure Oggi
INFORMATICA E PERSONAL COMPUTER
Informatica Oggi Settimanale • Informatica Oggi • Bit • PC Magazine • PC Floppy • Computergrafica & Applicazioni • Compuscuola • Trasmissione Dati e Telecomunicazioni
TECNOLOGIE E MERCATI
Watt • Lab News • Industria Oggi • Meccanica Oggi • Media Production • Strumenti Musicali
HOBBY E HOME COMPUTER • Fare Elettronica • Amiga Magazine • Amiga Transactor • Commodore Professional • Supercommodore 64 e 128 • Olivetti Prodest User • PC Software • PC Games • 3 ½ Software

In questa nostra tappa del viaggio all'interno dell'Amiga ci soffermeremo sul modo di lavoro dell'AmigaDOS. Per capire i cavilli dell'AmigaDOS dovete inanzitutto frenare l'impulso di sfasciare la tastiera e quindi evitare di riempire di cotone le fessure dei vostri drive.

I drive di cui è dotato l'Amiga sembrano essere molto più lenti di quello che non lo siano in realtà. L'hardware dedicato ai floppy disk può leggere un'intera traccia, circa 5600 byte, in un colpo solo, senza dare noie al microprocessore, tuttavia va detto che per quanto riguarda l'AmigaDOS, se compariamo il suo LIST con il normale DIR del PCDOS notiamo una notevole lentezza del primo rispetto al secondo. Perché? Semplice, il LIST non ha un posto dove andare per prelevare le informazioni necessarie inerenti al file mentre le directory del PCDOS contengono ben specificati i nomi dei file, la loro misura, e il luogo in cui si trovano.

L'AmigaDOS divide il disco da 3.5 in 1760 blocchi di 512 byte ciascuno, se volete lo potete verificare tramite il comando INFO del CLI. Undici blocchi comprendono una traccia (track), un giro completo. Il numero di blocco 880 è collocato nel mezzo della superficie del dischetto ed è il blocco principale (root block) che corrisponde alla root directory. Nel root block sono immagazzinati il nome del disco, l'ora e la data dell'ultima modifica, e la tavola di controllo (hash table).

L'hash table è una lista di puntatori alle subdirectory o blocchi di inizio file (file header blocks). L'AmigaDOS converte il nome del file in un numero intero positivo. Questo numero intero fa riferimento ad un'entratta nella hash table. L'entrata, a turno, è il numero del blocco contenente la testa del file (file header) o la subdirectory. Le subdirectory ed i file header assomigliano al blocco principale (root block); una subdirectory punta ai file header e alle altre subdirectory, mentre un file header punta ai blocchi di dati, dove il file è posto.

Per convertire i caratteri del nome di un file in numeri interi l'AmigaDOS applica una funzione di controllo particolare. Un esempio di questa funzione è A=1, B=2 eccetera. I problemi compaiono quando due nomi di file hanno lo stesso numero. Nell'AmigaDOS per esempio "Quake" e "Los Gatos" hanno tutti e due valore 14.

Per risolvere questo dilemma si ricorre alla subdirectory o al blocco dell'header del file. Questo blocco contiene la data e l'ora della sua creazione ed un puntatore indirizzato alla prossima header della quale nome ha lo stesso valore. L'AmigaDOS compara il nome nel blocco con l'altro. Se i nomi differiscono allora l'AmigaDOS passa al blocco successivo di header e così via, finché non trova il nome esatto o arriva alla fine della catena, in tal caso il file non esiste.

Questo meccanismo spiega la lentezza del comando LIST. L'AmigaDOS deve computare il nome del file, andare al prossimo file header, comparare i nomi, andare quindi al prossimo file header se non c'è un riscontro proseguire, e così via. Le informazioni del file sono praticamente disseminate invece di essere centralizzate come nel PCDOS.

Per alleggerire questo notevole ritardo, l'AmigaDOS mette le subdirectory ed i file header nella metà interna del disco ed i blocchi di dati nella metà esterna. Sfortunatamente però questo schema interno-esterno crea qualche difficoltà al Workbench. Quando voi aprite una drawer, l'Ami-

ALL'INTERNO DEL CLI

di Alessandro Prandi

Proseguiamo il nostro intinerario alla conquista del CLI

gaDOS deve portare la testina del drive all'interno per trovare il file header corretto e quindi all'esterno per leggere i file ".info", i quali contengono le icone. Questo rapido spostamento della testina del drive lungo la superficie del disco provoca un racappriciante gracidio. Fortunatamente le ultime versioni del Workbench raccolgono tutte le icone di un drawer in un singolo file.

Non si può negare che ad un primo esame il sistema di archiviazione si presenti perlomeno intricato, ma bisogna considerare anche i vantaggi che questo metodo offre. La misura e il numero di file sono limitati solo dalla capienza del disco. Ancora più importante, il sistema è sufficientemente sovrabbondante in modo da permettere che l'albero della directory possa essere ricostruito se una parte di esso viene distrutta. Due set di puntatori descrivono ogni branca dell'albero. Per esempio, gli header contengono i puntatori indirizzati al prossimo link nella catena vista prima, ai blocchi di dati e così via. Ogni blocco ha un numero in sequenza, il quale indica a chi appartiene, e un checksum il quale segnala se il blocco è stato danneggiato. Con la scansione dei blocchi, i numeri in sequenza, ed i puntatori un programma può ricostruire la struttura danneggiata del file del disco. Il Disk-Validator compie questo compito ma in proporzioni limitate. Ogniqualvolta inserite un dischetto il Disk-Validator controlla che la struttura di archiviazione sia conforme e che nessun blocco sia collocato due volte.

Per l'utente il sistema di archiviazione dell'AmigaDOS si presenta come una struttura convenzionale ad albero, percorribile tramite il comando CD. L'AmigaDOS vi offre alcuni comandi importanti per poter manipolare le informazioni contenute nelle varie header.

I blocchi della subdirectory e del file header comprendono una flag di protezione e lo spazio per un commento. Voi potete settare le flag di protezione con il comando PROTECT nel CLI o tramite l'Info del Workbench. Per proteggere un file dalla cancellazione scrivete:

PROTECT <nomefile> rwe

Per permettere la cancellazione di un file non cancellabile scrivete:

PROTECT <nomefile> rwed

Le future versioni del software di sistema permetteranno anche di proteggere il file dalla lettura, scrittura e dall'esecuzione.

Per aggiungere un commento ad un file scrivete:

FILENOTE <nomefile> "testo"

dove il testo può arrivare sino ad 80 caratteri. Il commento lo potete vedere LISTando il file. Voi potete vedere o cambiare il commento anche tramite l'Info del Workbench. Con il comando COPY non si copia il commento del file, poiché questo comando crea un nuovo header del file.

Quando voi premete due volte il tasto del mouse su un tool o scrivete il nome del file da CLI l'AmigaDOS ricerca il file. Quando il sistema di archiviazione lo localizza il caricatore lo recupera. Il loader legge il programma nella memoria usando una tecnica chiamata scatter loading (caricamento frazionato). Il caricatore fraziona il file in pezzi chiamati hunk (fette). Ogni fetta comprende l'informazione di come essa è inserita rispetto alle altre. Il loader a questo punto ripone ciascuna fetta, ogniqualvolta riesce a trovare spazio nella RAM e le modifica in modo che ne possano trovare un'altra. Lo scatter loading sfrutta al meglio la memoria disponibile.

Una volta che il caricatore ha finito il suo compito l'AmigaDOS deve definire come il programma interagisce con il resto del computer. Il DOS infatti tratta il programma come un processo. Ogni processo pensa ad un proprio uso esclusivo del microprocessore, anche se altri processi possono spartire l'uso del processore.

I programmi caricati da CLI condividono il processo CLI, mentre il Workbench genera un nuovo processo per ogni tool clickato due volte. L'AmigaDOS inoltre crea dei processi per sorvegliare la porta seriale, il drive, la porta parallela e così via.

Questi vari tipi di processi comunicano tra loro attraverso delle porte di comunicazione che agiscono similarmenta ai nostri telefoni. Per esempio un editor di testo può chiedere al processo del disco di leggere in un documento. L'I/O del disco però rimane impantanato quando due processi richiedono il disco allo stesso momento. La testina del drive vola avanti ed indietro prelevando i dati di un processo e quindi dell'altro. Dal punto di vista dell'utente i processi sembrano agire contemporaneamente. Diversi comandi vi permettono di esaminare e controllare questi processi.

Normalmente i programmi condividono il task del CLI ma voi potete usare un prefisso per il nome del file con il comando RUN, con questa istruzione l'AmigaDOS esegue il programma come un processo indipendente. Voi potete usare una stringa di programmi usando il segno più:

RUN COPY File RAM: +
DELETE File +
ECHO "File spostato in RAM"

I processi generati con il RUN continuano ad usare la finestra CLI per l'output, per creare un nuovo processo con una sua finestra scrivete NEWCLI. Con il comando STATUS potete listare quali processi CLI sono in esecuzione.

L'AmigaDOS non può interrompere un processo una volta iniziato, essi devono estinguersi per conto loro. Molti programmi tuttavia vi concedono molto gentilmente di essere fermati durante l'esecuzione usando i tasti CTRL-C. Al posto di CTRL-C potete anche usare il comando BREAK, BREAK 2 interomperrà l'esecuzione nel secondo CLI task, quello con il segno "2>". Quello che succede quando usate CTRL-C o BREAK dipende solamente dal programma, un programma che si rispetti riconosce il segnale di interruzione e si ferma.

Benché l'AmigaDos offra all'utente dei mezzi molto potenti non si può definire il suo ambiente dei più amichevoli. Fortunatamente il programmatore può accedere ai compiti dell'AmigaDOS modificando alcune funzioni. In effetti ci sono già due programmi in grado di addomesticare il CLI, questi sono lo SHELL e MyCLI.

Ad esempio il programma SHELL scritto da Randell E. Jesup traduce i vostri sinonimi in comandi AmigaDOS. Basterà inserire un file "comandi" nella directory SYS:. Il file contiene la risistemazione dei comandi CLI. Un solo sinonimo può comprendere un'intera lista di comandi CLI dandovi delle notevoli capacità macro. Comunque una volta che lo SHELL ha compiuto i suoi processi passa i comandi attraverso al CLI: essi non sono comandi interni.

Indubbiamente l'integrazione dell'AmigaDOS da parte di programmi di questo genere semplifica e sopratutto velocizza enormemente il lavoro da eseguire mediante CLI. Un'altro aspetto importante di questi programmi è il tipo di editing: nel CLI spesso dopo aver scritto una lunga sequenza di comandi vi accorgete di aver commesso un'errore all'inizio della riga (parolaccia!) e quindi siete costretti a cancellare e riscrivere quasi l'intera sequenza. L'editor di questi programmi invece si comporta quasi come quello di un normale wordprocessor con talvolta delle varianti molto interessanti.

Sicuramente nei prossimi numeri assieme al CLI tratteremo parallelamente anche i modi di impiego di programmi molto utili come ad esempio lo Shell della Metacomco.

di Giorgio Dose

COME AUMENTARE DEI PROGRAMMI

SPEED

Avete appena scritto un programma in Basic sul vostro Amiga e lo trovate un po' lento? Volete aumentarne il numero di giri? Seguendo le tecniche usate dai compilatori e spremendo un po' le vostre meningi è possibile incrementare considerevolmente la velocità di esecuzione.

Esistono i metodi più svariati per migliorare un programma e renderlo più veloce ed ognuno di noi conosce diversi trucchetti per farlo. Quelle che troverete di seguito sono delle regole generali che molti già conoscono ma che possono tornare utili ai programmatori meno esperti. Alla fine dell'articolo inoltre sono riportati alcuni programmi per valutare i vantaggi che si ottengono usando queste tecniche.

Cosa fanno i compilatori

I vari compilatori C, Fortran, Pascal usano varie tecniche per diminuire i tempi di esecuzione delle loro routine. Molti di essi esaminano il codice oggetto e individuano dove e come va ottimizzato. Alcuni di essi analizzano i passi in cui il programma impiega molto tempo e le variabili poste all'interno dei loop, poi decidono con "intelligenza" i cambiamenti da apportare al programma stesso. Altri compilatori invece fanno le loro modifiche solo dopo aver creato il linguaggio macchina.

Le tecniche che verranno descritte di seguito possono essere applicate a quasi tutti i linguaggi a livello sorgente.

Il metodo più semplice è conosciuto come "costant folding". Il compilatore esamina tutte le costanti che possono essere combinate fra loro in modo da eliminare i relativi calcoli durante l'esecuzione. Considerate la seguente funzione:

```
x = 11 + 2 * (x * 9/6 + y) * 8.6 + 4 * 5
```

dopo che il compilatore ha eseguito la trasformazione delle costanti diventerà:

```
x = 17.2 * (x * 1.5 + y) + 31
```

Sono state eliminate due moltiplicazioni, una divisione e una addizione ed il tempo di esecuzione risulterà notevolmente più breve.

Un altro metodo molto noto è il cosiddetto "code motion" e consiste nel ridurre il numero di operazioni matematiche all'interno di un loop. Il compilatore esamina quali espressioni non vengono mai modificate all'interno di un loop e quindi porta le stesse al di fuori del ciclo in modo da evitare la ripetizione dei calcoli. Consideriamo ad esempio il seguente loop:

```
FOR I = 1 TO 360 STEP 10
   FORJ = 0 TO 360 STEP
      R = 20 + W
      K = SIN(I)*SIN(J)+COS(I)*COS(J)
      Z=Z+J+R
  NEXT J
NEXT I
```

La prima funzione che salta all'occhio è ovviamente R = 20 + W; non c'è nulla infatti, all'interno del loop, che possa modificare il valore di R o di W. Il compilatore allora porta l'espressione fuori da entrambi i cicli FOR-NEXT.

Osservando il programma con maggio-

LA VELOCITÀ IN AMIGABASIC

BASIC!

re attenzione si può notare che anche la variabile I rimane costante nell'ambito del loop interno e così pure il valore di SIN(I) e COS(I). È quindi possibile portare anche questi due calcoli al di fuori del loop. Il risultato ottimizzato sarà quindi:

```
R = 20 + W
FORI = 1 TO 360 STEP 10
    X1 = SIN(I)
    X2 = COS(I)
    FOR J = 1 TO 360
        K = X1 * SIN(J) + X2 * COS(J)
        Z = Z + J + R
    NEXT J
NEXT I
```

L'aver portato fuori dal ciclo due funzioni come SIN e COS incrementa significativamente la velocità di esecuzione del programma perché riduce notevolmente il numero di calcoli eseguiti. Per ottenere questo miglioramento il compilatore ha dovuto generare due proprie variabili interne X1 e X2 e rimpiazzare con esse le funzioni SIN(I) e COS(I). Da rilevare che l'operazione di cui sopra aumenta le prestazioni ma occupa più memoria.

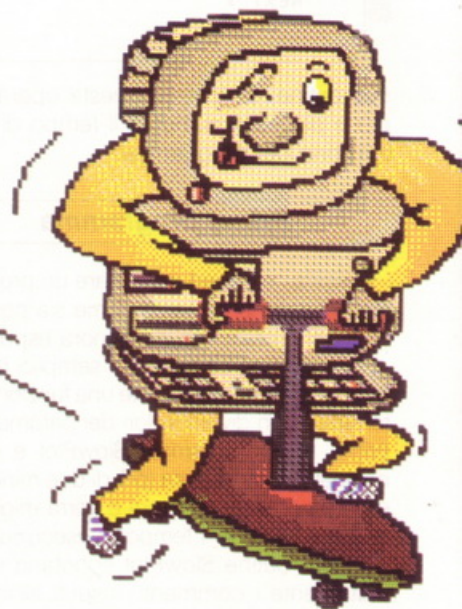
Veniamo al BASIC

I metodi visti or ora possono venire efficacemente usati nei vostri programmi in AmigaBasic. L'AmigaBasic è un linguaggio interpretato e come tale non è in grado di ottimizzare da solo i propri programmi; siamo noi che dobbiamo decidere come e dove intervenire nei vari passi del processo. Per scrivere programmi più veloci è bene rispettare queste semplici regole:

1) Non usare variabili quando è possibile usare delle costanti. Ogni volta che l'interprete Basic esamina una variabile deve guardare la sua tabella delle variabili per trovarne il valore. Questo non richiede molto tempo, però se la cosa si ripete molto spesso la velocità scende rapidamente.

2) Non inserire commenti nei loop che richiedono molto tempo per essere completati. L'esame dei commenti normalmente non comporta dei ritardi apprezzabili al programma ma nei cicli FOR-NEXT aumenta senz'altro il tempo di attuazione dei loop stessi. Questo non è un invito a non inserire mai i commenti ma solo un consiglio a non metterli nei punti più critici. I commenti infatti sono molto utili perché facilitano la lettura del programma e la sua revisione o modifica anche diverso tempo dalla stesura ed inoltre consentono ad altre persone di comprendere meglio il funzionamento del programma stesso.

3) È importante conoscere tutte le funzioni dell'AmigaBasic; può sembrare una cosa ovvia ma è molto facile dimenticarsi delle funzioni meno comuni e creare al loro posto delle routine meno efficienti.

4) Evitare le chiamate frequenti alle subroutine. Se ad esempio un programma chiama la stessa routine ripetutamente all'interno di un loop è consigliabile inserire il codice della subroutine all'interno del ciclo. Se le subroutine sono più d'una bisogna cercare, se possibile, di fonderle insieme e crearne una sola. Ogni volta che viene chiamata una subroutine infatti l'AmigaBasic mantiene traccia del punto in cui deve ritornare dopo aver eseguito la

```
'Listato 1
'Migliorare la velocita' dei programmi
'
'Esempio di "plotting"
'
DEF FNa(t) = (100/360*t)/3
'
'Disegna un punto alla volta e
'riporta il tempo impiegato
'
CLS
GOSUB SlowPlot
FOR i = 1 TO 10000: NEXT i

CLS
GOSUB QuickPlot
FOR i = 1 TO 10000: NEXT i

CLS
GOSUB QuickerPlot

LOCATE 1,1
PRINT "SlowPlot:    Begin -- ";a1$;" End - ";a2$
PRINT "QuickPlot:   Begin -- ";b1$;" End - ";b2$
PRINT "QuickerPlot: Begin -- ";c1$;" End - ";c2$
END

SlowPlot:
PRINT "SlowPlot"
a1$ = TIME$
FOR j = 20 TO 100 STEP 20
   FOR x = -150 TO -1 STEP .2
'
'     Prima disegna la parte destra
'
      y = SIN(FNa(x))/(FNa(x))*j+50
      PSET (x+150,y),1
   NEXT x
   FOR x = 1 TO 150 STEP .2
'
'     Ora disegna la parte sinistra
'
      y = SIN(FNa(x))/(FNa(x))*j+50
      PSET (x+150,y),1
   NEXT x
NEXT j
a2$ = TIME$
RETURN

QuickPlot:
PRINT "QuickPlot"
b1$ = TIME$
FOR j = 20 TO 100 STEP 20
   FOR x = 1 TO 150 STEP .2
      sy = x*.0925926
      y1 = SIN(sy)/sy*j+50
      PSET (150-x,y1),1
      PSET (x+150,y1),1
   NEXT x
NEXT j
b2$ = TIME$
RETURN

QuickerPlot:
PRINT "QuickerPlot"
c1$ = TIME$
FOR x = 1 TO 150 STEP .2
   sy = x*.0925926
   SinTemp = SIN(sy)/sy
   FOR j = 20 TO 100 STEP 20
      y1 = SinTemp*j+50
      PSET (150-x,y1),1
      PSET (x+150,y1),1
   NEXT j
NEXT x
c2$ = TIME$
RETURN
```

subroutine. E ripetere questa operazione più e più volte aumenta il tempo di effettuazione del programma.

Primo programma

Passiamo ora ad esaminare un programma esempio e vediamo come sia possibile mettere in pratica quanto finora esposto.

Il listato uno contiene tre semplici routine per disegnare cinque volte una funzione matematica con diversi valori dei parametri. La prima routine chiamata SlowPlot è quella originale, è la più lenta (più di due minuti per essere portata a termine) e verrà migliorata al fine di ridurne il tempo di esecuzione.

Nella routine SlowPlot si notano immediatamente i commenti, inseriti all'interno dei loop, che vanno eliminati o portati all'esterno del ciclo, e la funzione FN che può essere tolta e sostituita da una semplice variabile.

Osservando poi sullo schermo il disegno della funzione si rileva che è simmetrico rispetto al centro; possiamo quindi trarre vantaggio da questa simmetria, unire i loop interni e disegnare entrambi i lati della funzione nello stesso tempo.

Le variazioni apportate alla routine SlowPlot si evidenziano nella successiva routine QuickPlot. QuickPlot impiega circa 50 secondi per essere condotta a termine, un buon risultato!

Ma vediamo come si possa ancora migliorare...

Nell'esecuzione del loop più interno della routine QuickPlot si nota che solo l'espressione contenente la variabile j presenta un valore sempre diverso, mentre i calcoli con lo stesso valore della variabile x vengono ripetuti cinque volte (una per ogni diverso valore di j). Invertendo il loop interno con quello esterno, per ogni punto della funzione disegnato sullo schermo, i calcoli, comprendenti la variabile x, vengono eseguiti una sola volta e non cinque volte come avveniva in precedenza.

Con le ultime variazioni siamo arrivati all'ultima routine QuickerPlot che si completa in circa 40 secondi. Possiamo ben dire di aver fatto un buon lavoro.

Le tre routine descritte sono state inserite nel programma una di seguito all'altra in modo che si possa facilmente valutare il diverso rendimento di ognuna.

Secondo programma

La tecnica di ottimizzazione che andiamo a presentare è senzaltro la più efficace ma anche la più difficile. Dopo aver com-

pletato un programma e aver fatto il possibile per aumentare le sue prestazioni, facciamo mentalmente un passo indietro e chiediamoci: le routine che abbiamo scritto sono veramente le migliori o potevamo fare di meglio?

È molto difficile dare una risposta e lo è ancor di più per i programmatori meno esperti. Possiamo fare un piccolo test, non per valutare le nostre capacità ma bensì per imparare, da un semplice esempio, che non sempre la soluzione più logica è la migliore, anzi...

Provate ad esempio a scrivere una routine per mescolare un mazzo di carte, supponendo di avere un array già inizializzato con il nome delle carte e di poter usare qualsiasi altra variabile. Avete fatto? Bene, potete continuare nella lettura.

Molti programmatori, in un caso come questo, avrebbero scritto una routine molto simile alla routine ShuffleA del listato numero due. L'idea di base è di creare, oltre a quello contenente i nomi delle carte, due nuovi array, uno per mettervi le carte mescolate man mano che vengono selezionate ed uno per segnare quelle che sono già state prelevate (non vogliamo avere duplicati).

La routine inizialmente pulisce l'array Check o meglio lo inizializza completamente a zero. Poi, all'interno del ciclo FOR-NEXT, usa un numero random per selezionare una carta a caso e controlla che la carta non sia già stata estratta guardando nell'array Check. Se era già stata scelta si preleva un'altro numero random finché non si trova una carta che ancora non era stata toccata; a questo punto si pone quest'ultima nell'array Shuffled. Il ciclo si ripete finché tutte le 52 carte sono state selezionate e poste a caso nell'array.

Questo può inizialmente sembrare un buon algoritmo ma esaminandolo attentamente si nota che esso potrebbe impiegare molto tempo per estrarre tutte le carte del mazzo. Questo perché man mano che le carte vengono scelte e posizionate nel nuovo array, aumentano le probabilità che esca un numero random corrispondente ad una carta già mescolata, con conseguente ripetizione del ciclo.

Per rimediare a questo difetto, se così possiamo chiamarlo, è necessario inserire nella routine un certo livello di intelligenza artificiale. È quanto si ècercato di fare nella seconda routine chiamata ShuffleB. Questa routine usa meno memoria della precedente e risulta più veloce nell'esecuzione. L'idea consiste nel prelevare, all'interno di un ciclo FOR-NEXT, un numero random compreso tra uno e 52 e quindi nello scambiare di posto, nell'array contenente il nome delle carte, la carta corrente (avente il numero di posizione progressivo uguale alla variabile I del loop) con quella il cui numero corrisponde al numero random. Così facendo vengono sempre prelevati solo 52 numeri random e la routine impiega sempre lo stesso tempo per mescolare tutte le carte.

Per mostrare con maggior evidenza la diversa velocità di esecuzione delle due routine il programma del listato due inizializza e mescola il mazzo di carte per ben dieci volte. La routine A impiega da 18 a 22 secondi per essere portata a termine mentre la routine B si completa sempre in otto secondi.

Spero che gli esempi sopra riportati siano di stimolo a migliorare sempre i propri programmi; non fermarsi mai alla prima idea che viene in mente ma tentare continuamente nuovi e diversi modi di risolvere un problema, può sempre capitare di inciampare in una soluzione migliore e più veloce.

```
'Listato 2
'
'Migliorare la velocita' del programma
'
'Esempio "Come mescolare un mazzo di carte"
'
DIM card$(52), Shuffled$(52), Check(52)
DIM Spot$ (4)

Spot$(0) = " di cuori"
Spot$(1) = " di quadri"
Spot$(2) = " di fiori"
Spot$(3) = " di picche"
DATA "asso","due","tre","quattro","cinque","sei","sette"
DATA "otto","nove","dieci","jack","donna","re"

Main:
RANDOMIZE TIMER
'
' ShuffleA
'
PRINT "Shuffling ..."
a1$ = TIME$
   FOR S = 1 TO 10

      RESTORE
      GOSUB Init
      FOR I = 1 TO 52
         Check(I) = 0
      NEXT I

      FOR I = 1 TO 52
Again:
         X = INT(RND*52+1)
         IF (Check(X) = 1) GOTO Again
         Shuffled$(I) = card$(X)
         Check(X) = 1
```

```
   NEXT S
a2$ = TIME$
PRINT "ShuffleA -- Start: ";a1$;" End ";a2$
'
'ShuffleB
'
PRINT "Shuffling ..."
b1$ = TIME$
   FOR S = 1 TO 10

      RESTORE
      GOSUB Init
      FOR I = 1 TO 52
         Check(I) = 0
      NEXT I

      FOR I = 1 TO 52
         X = INT(RND*52+1)
         SWAP card$(I),card$(X)
      NEXT I
   NEXT S
b2$ = TIME$
PRINT "ShuffleB -- Start: ";b1$;" End ";b2$
END

Init:
X = 1
FOR J = 1 TO 13
   READ A$
   FOR I = 0 TO 3
      card$(X) = A$+Spot$(I)
      X = X + 1
   NEXT I
NEXT J
RETURN
```

INFORMATICA PROFESSIONISTI

Mauro Risani

I COMANDI DI LOTUS 1-2-3

REFERENCE GUIDE

pp. 96 Lire 12.500
Cod. 051T

Un'utile e veloce reference concernente tutti i comandi del popolare foglio elettronico della Lotus, indispensabile per consultare velocemente la sintesi di un dato comando, di una funzione, di una procedura macro.

James Cavuoto / Jesse Berst

VENTURA

IL GRANDE MANUALE

pp. 410 Lire 55.000
Cod. PP593

Una guida completa ed esauriente che si rivolge a chiunque voglia imparare a usare Ventura Publisher nella produzione di riviste, libri, manuali, documentazione, proposte, modulistica.

PERSONAL COMPUTER

Davide Pandini

LINGUAGGIO C

REFERENCE GUIDE

pp. 116 Lire 12.500
Cod. R671

La trattazione delle funzioni della libreria standard prendendo in considerazione lo standard ANSI ma riportando comunque le differenze di implementazione relative al sistema operativo UNIX e al "K.&R. standard", indicandole come eccezioni.

ELETTRONICA CONSUMER

Amadio Gozzi

77 SCHEDE PER IL RIPARATORE TV

FUNZIONAMENTO E RIPARAZIONE

pp. 330 Lire 40.000
Cod. BE718

Una vera guida operativa per il tecnico TV, in cui viene messa a disposizione tutta l'esperienza raggiunta dal suo autore sia attraverso l'attività di laboratorio, sia attraverso l'insegnamento tecnico pratico effettuato presso il Ceniart (Centro per l'Informatica e l'Assistenza Radio TV).

PERSONAL COMPUTING

Robert Krumm

MS DOS ADVANCED

IL MANUALE DEL PROGRAMMATORE

pp. 426 Lire 55.000
Cod. R600

Dopo aver chiarito i concetti fondamentali del sistema operativo su disco (DOS), vengono discussi i programmi di utility, i più noti programmi di potenziamento della tastiera e le utility di background, fornendo al lettore la capacità di ottenere il massimo dai suddetti programmi.

INFORMATICA PROFESSIONALE

Valerio Alessandroni/Massimo Calda

ROBOTICA

FONDAMENTI E APPLICAZIONI

pp. 256 Lire 38.000
Cod. GE584

Un'ampia descrizione del mondo dei robot industriali, attraverso l'hardware (l'aspetto meccanico), il software (il linguaggio), le periferiche (sensori ed attuatori), le modalità di selezione ed impiego ed i criteri di sicurezza.

Ted J. Biggerstaff

SOFTWARE DI BASE

STRUMENTI DI SVILUPPO

pp. 392 Lire 52.000
Cod. GY629

Un'efficace ed ampia spiegazione su come modernizzare ed estendere il sistema operativo di un personal computer IBM o compatibile, svelando i segreti della programmazione e mettendo in grado il lettore di sviluppare un nuovo sistema operativo molto più potente ed attuale.

IL TUO LIBRO.

GEJ 0063

Per acquistare libri Jackson rivolgetevi alle migliori librerie e negozi di informatica oppure utilizzate l'apposito tagliando riportato in fondo alla rivista.

- **Disco Magazine Forever**
- **GIOCHI**
  - Blackjack
  - Labirinto
- **MUSICA**
  - Music
  - Laboratorio del Suono
- **STRUMENTI**
  - Dirutility
  - Lente
- **MAGAZINE**
- **GRAFICA**
  - Palette
  - BasicBoing
  - SeeILBM
  - Astronave
  - Guerre stellari
- **WITZ**
  - Sproing
  - Boing
  - Ombre

DISCO MAGAZINE FOREVER

Attenzione! Attenzione! Siamo il Fronte Combattente del Chip Democratico e dirottiamo questa astronave sul pianeta Disco Magazine II

di Alessandro Prandi

Mi trovavo alla guida della Trans-Istor, astronave di raffinata tecnologia appartenente alla flotta multigalattica MBI, nell'anno terrestre 1988, quando sul diario di bordo dovetti annotare i seguenti avvenimenti.

Fase solare: 4. Distanza dal pianeta Centrale: leghe MBI 120. Presunto arrivo: fase solare 6.

Ormai poco meno di due fasi solari separavano me e i passeggeri della Trans-Istor da un meritato riposo nelle lussuose sale messe a disposizione dalla MBI a tutti i clienti e equipaggi delle sue linee interplanetarie.

Fase solare: 5. Distanza dal pianeta Centrale: leghe MBI 60. Presun....

A questo punto irruppe nella cabina di comando un gruppo di tre umanoidi e un replicante, i quali senza troppi complimenti, mi comunicarono un messaggio da annunciare ai passeggeri. Con un disintegratore atomico puntato alla gola lessi le seguenti parole: "Attenzione! Attenzione! Siamo il Fronte Combattente del Chip Democratico e dirottiamo questa astronave sul pianeta Disco Magazine II."

Da quel tempo sono passate innumerevoli fasi solari e, sinceramente, debbo dire che nè io nè i miei passeggeri d'allora invidiamo le piatte forme di vita del pianeta Centrale. Le giornate su Disco Magazine II trascorrono meravigliosamente e ogni tanto ricordiamo con gioia che fortunatamente nessun riscatto è stato pagato per il nostro ritorno.

Città di Amiga, anno 5034

Storie come questa ormai non fanno più notizia da quando l'Amiga è entrato nelle nostre case. La sua capacità di acquisire proseliti ovunque è ben nota anche ai più disinteressati. Avremmo potuto stupirvi con effetti speciali, invece la realtà ci ha imposto una scelta obbligata ed insostituibile nel guidarvi alla conoscenza completa dell'universo Amiga. Questa scelta è stata fatta! Amiga Magazine è nata e cresce di giorno in giorno, a testimoniare che chiunque si dedichi a questa macchina con convinzione e professionalità viene largamente ricompensato.

Solo un anno fa, lo scetticismo nei confronti di questo prezioso ed inimitabile mezzo regnava in molti di voi, mentre oggi sicuramente rimpiangete il tempo sprecato nell'esitazione. Le fonti da cui attingere informazioni e programmi su Amiga sono ormai praticamente innumerevoli e sopratutto inesauribili, e ciascuna nel profondo nasconde un vaso di Pandora. Il nostro obiettivo, non lo nascondiamo, è quello di diventare la vostra fonte principale, ma soprattutto di essere il tramite della vostra conoscenza, in modo da poterla trasmettere a tutto il mondo Amiga.

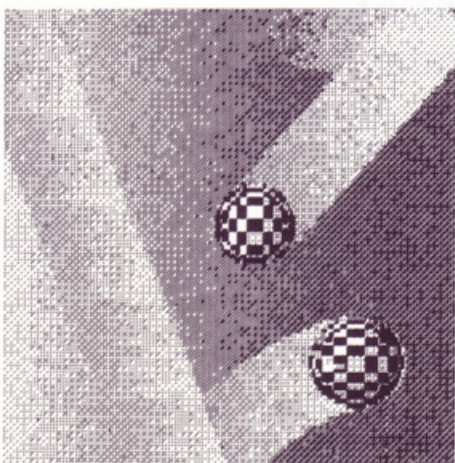
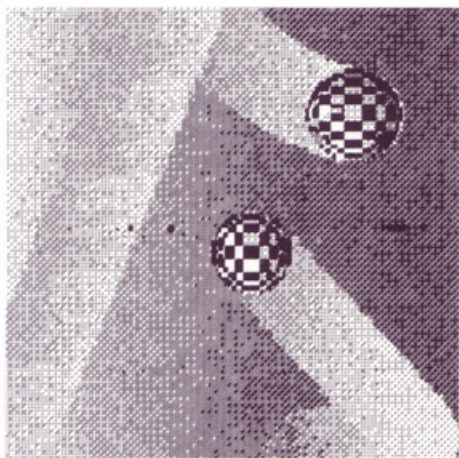
Non ci riteniamo santoni o profeti o presentatori alla "vù cumprà", ma semplicemente, gente che come voi condivide le gioie (molte) e le pene (poche) dell'evento Amiga.

L'astronave atterrò su Disco Magazine II e da quel momento divenimmo testimoni oculari di un meraviglioso universo!!

La struttura del disco si snoda nella praticissima configurazione ad albero e vi offre sei directory principali nelle quali troverete i programmi relativi a ciascun argomento. Le directory sono:

Witz
Giochi
Musica
Grafica
Strumenti
Magazine

Dopo aver sperimentato con successo nel primo numero questo tipo di gestione del dischetto, siamo dunque giunti alla conclusione di aver imboc-

cato la strada giusta e più facilmente percorribile.

Witz è la directory delle sorprese e delle curiosità, in essa troverete ogni volta dei simpatici programmi che faranno divertire voi e il vostro computer. Assieme ai programmi eseguibili troverete sempre i listati sorgente, in modo da stimolare chi di voi volesse apportare delle modifiche o miglioramenti. Naturalmente questa directory non racchiuderà mai dei listati di irraggiungibile levatura tecnica, ma comunque vi fornirà interessanti spunti su cui riflettere.

Passiamo ora al piatto più ghiotto per gli Amiga-player. Nella directory dei Giochi potrete misurare il vostro grado di abilità, perseveranza, e, talvolta di sola fortuna. Per gli Amigomani più incalliti promettiamo fin d'ora una rotta costellata di arcade, adventure e chi più ne ha più ne metta. Nell'immediato prossimo futuro, infatti, inseriremo in questa parte del disco delle chicche di pregevole valore. Lasciateci solo il tempo di confezionarle in modo che siano degne di voi.

Musica è la directory dedicata a quelli che credono di poter ancora imparare qualcosa dal Direttore d'orchestra Amiga. Naturalmente sono ben accetti anche i rock fans.

Ed ecco qualcosa che forse vi coinvolgerà all'unisono, la Grafica!! A questo lato così seducente della personalità Amiga, dedichiamo il massimo spazio consentitoci. In questo numero vi sarà possibile, oltre ai vari programi, ammirare dei suggestivi "schizzi" espressi con il programma De Luxe Paint.

Se qualcuno di voi ha già avuto la fortuna di conoscerci, avrà sicuramente apprezzato la directory Strumenti, dedicata alle varie utility necessarie nel lavoro quotidiano con Amiga. La ricerca in questo campo è veramente vasta, ma il reperimento di programmi apprezzabili è tutt'altro che facile, comunque, siamo convinti di potervi servire ora ed in futuro solo il meglio.

L'ultima, e forse per i più svogliati, anche la più attesa, è la directory Magazine. Qui troverete tutti i listati, (FUNZIONANTI!), contenuti nella rivista. Per darvi maggior sicurezza e per evitare che il vostro lavoro venga reso vano da uno stupido, ma ahimè possibile, errore nella fase di stampa dei listati, abbiamo pensato di accludere i programmi contenuti nella rivista, nel relativo dischetto.

Convenzioni presenti e future. Sì, anche noi vogliamo e dobbiamo segnalarvi alcune regole da seguire per una migliore comprensione del contenuto del dischetto. Analizzando le varie directory e sottodirectory da CLI noterete che alcuni file comprendono delle estensioni che, a seconda dei diversi linguaggi, riguardano i relativi file sorgente. Se il nome di un programma non comprende alcuna estensione, vorrà dire che esso è direttamente eseguibile, basterà digitarne il nome, preceduto eventualmente dal percorso (drive/directory...). I file contenenti l'estensione ".info" sono necessari al sistema operativo per raffigurare le varie icone quando si adopera il Workbench. Illustrati questi concetti, passiamo ora ad un esame più dettagliato delle varie estensioni che potrete incontrare nei dischetti di Amiga Magazine.

.bas	sorgente in Basic
.c	sorgente in C
.h	sorgente header in C
.asm	sorgente in Assembler
.def	sorgente modulo definizione in Modula-2
.mod	sorgente modulo implementazione in Modula-2
.f	sorgente Forth
.scr	sorgente screen Forth
.o	file oggetto non linkato
.obj	file oggetto non linkato
.lsp	sorgente in Lisp

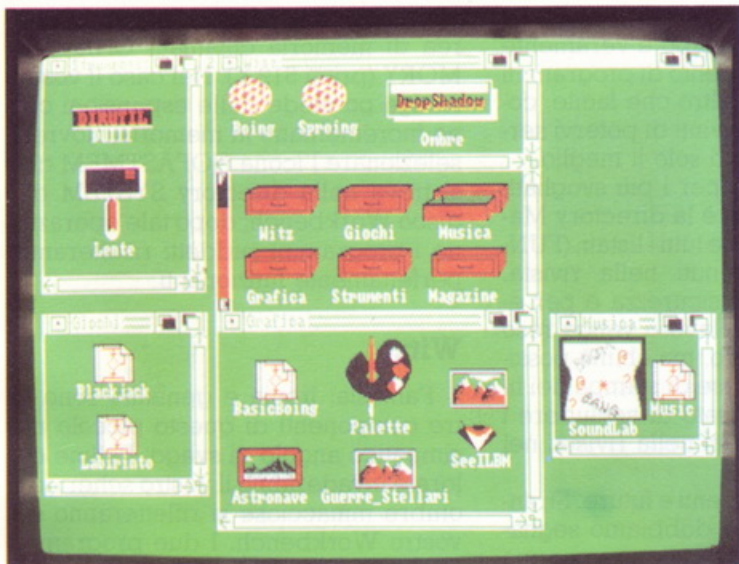
Le estensioni appena elencate ovviamente sono quelle che noi pensiamo di adoperare più frequentemente, per cui nel caso dovessimo offrirvi dei programmi di diversi linguaggi da quelli appena detti, sarà nostra premura informarvi delle relative variazioni.

Un'altra considerazione di primaria importanza riguarda quei programmi che, una volta caricati, occupano l'area di memoria chiamata CHIP MEMORY (primi 512 K). Nel caso il vostro Amiga possieda delle espansioni che ne incrementano la memoria, dovrete selezionare l'icona NOFASTMEM che si trova nella directory SYSTEM del disco Workbench; dopo tale operazione i programmi suddetti risulteranno perfettamente funzionanti.

Witz

Fantasia, ironia e genialità sono le tre componenti di questo piccolo ma simpatico angolo di svago. Palline colorate invaderanno il vostro schermo e ombre minacciose si rifletteranno sul vostro Workbench. I due programmi Sproing e Boing, una volta lanciati, faranno rimbalzare davanti ai vostri occhi delle palline bianco-rosse. Noterete che contemporaneamente alla loro comparsa viene visualizzata una mini-finestra, per interromperne l'esecuzione basterà portarsi nel gadget sinistro e premere il tasto sinistro del mouse. Per quanto riguarda il listato sorgente in C di Boing dovete tener presente che esso e studiato per il compilatore Atzec e che quindi lavora su 16 bit, essendo sicuri della vostra profonda conoscenza in materia vi lasciamo la libertà di apportare le modifiche necessarie per la conversione in un file sorgente che possa essere compilato dal Lattice. Ricordate che Sproing NON può essere lanciato da CLI! Ombre è invece un programma d'effetto nel vero senso della parola. Il programma, eseguibile anche da CLI, dà un ombra a tutte le finestre presenti sullo schermo, dandovi così l'illusione di lavorare in un ambiente tridimensionale. Nella finestra aperta dal programma trovate due indicatori regolabili tramite mouse, uno controlla l'intensità dell'ombra mentre l'altro regola la distanza dalla luce. Anche qui per interrompere l'esecuzione dovete portarvi sul gadget sinistro della finestra e premere il tasto del mouse. Purtroppo non possiamo fornirvi i listati dei programmi Sproing e Ombre come promesso. Se non vi è troppo difficile, provate a perdonarci.

<<Non donar loro nulla. Piuttosto togli loro qualche cosa od aiutali a portarla; ciò recherà ad essi qualche sollievo...>>.

F. NIETZCHE

Menu di Disk Magazine 2

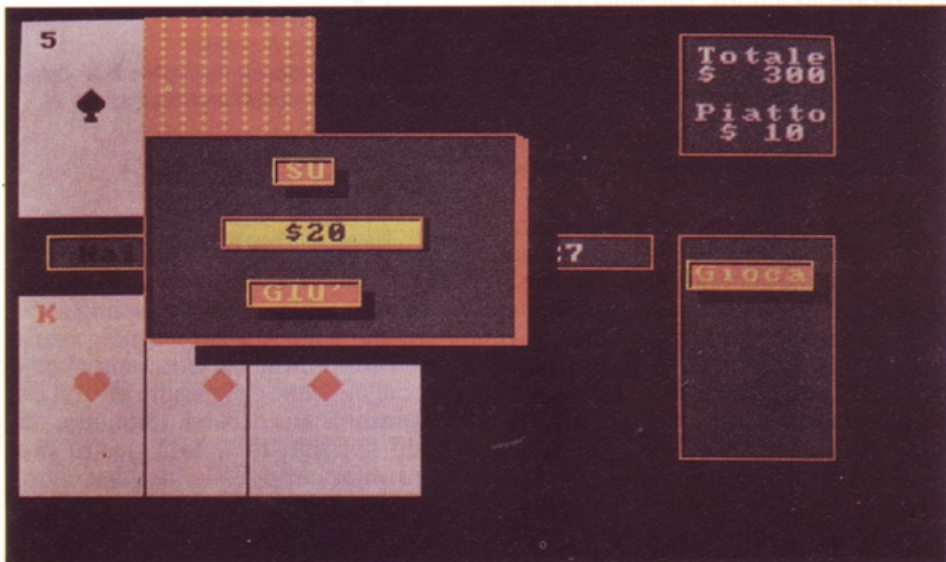
Giochi

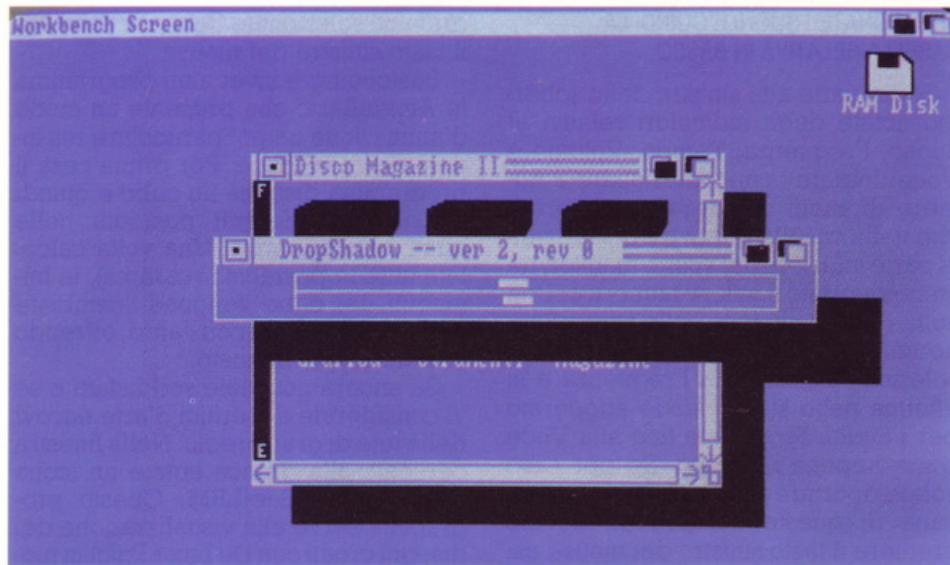
Blackjack e Labirinto sono i giochi con i quali vi misurerete questo mese. Il gioco d'azzardo è dunquè arrivato anche nel vostro Amiga. Con Blackjack vi sembrerà di vivere le emozioni dei cow-boy americani quando sfidano le slotmachine dei casinò di Las Vegas. Il gioco, dopo la schermata iniziale mescola le carte e ne dà due a voi e due al computer, la seconda carta del computer è coperta, a questo punto dovete cercare di arrivare a 21. Dopo avervi fornito le prime due carte un messaggio vi indicherà il punteggio, (un 5 e un 8 daranno 13 e così via); a seconda di questo punteggio potrete richiedere ancora delle carte. Ovviamente se la somma dà un numero maggiore di 21 avete perso. Se siete soddisfatti della smazzata potete clickare con il tasto sinistro del mouse la parola Resta, se invece desiderate ricevere ancora una carta dovete selezionare Vai. Sotto Vai e Resta trovate scritto 'x 2' questo significa che potete raddoppiare la posta in gioco. Oltre alle opzioni illustrate fin qui c'è anche un menu a discesa selezionabile con il tasto destro del mouse. A sinistra troviamo l'Esci che vi porterà fuori dal programma. Opzione dedicata ai più perfidi è quella di Truffa. Clickando su Truffa si aprirà una finestra sopra la scritta Vai e qui, spregevoli bari, potrete leggere il punteggio del computer a carte coperte. L'ultimo menu vi permette invece di gonfiare il vostro portafoglio che inizialmente è di 300 Dollari, selezionando Totale nel menu Cambio potrete arrivare fino ad un bottino di 5000 $. Scegliendo invece Puntata potete incrementare il piatto sino a 500 $ per smazzata. Ricordate che le figure come Jack, Donna e Re valgono 10 mentre il valore dell'Asso dipenderà dal suo abbinamento con una o l'altra carta, infatti l'asso può valere sia uno che undici, per esempio A e K valgono 21, mentre 6 e A fanno 7. Il croupier, come è ovvio, lo fa il computer pertanto in caso di parità vince il banco ovvero il vostro incallito Amiga. Ora che sapete tutto sul Blackjack potete iniziare la sfida, ma attenti, i vostri drive potrebbero impazzire e regalarvi un fiume di denaro!!

Labirinto è invece un gioco che metterà a dura prova i vostri nervi. Anche questo come Blackjack è scritto in Basic ma le sue dimensioni sono notevolmente inferiori. Quando inizia l'esecuzione il programma crea un labirinto con l'accompagnamento di sinistri rumori. Una volta terminato il disegno un punto giallo comparirà sulla sinistra, vicino alla porta d'entrata del Labirinto. Sul lato destro noterete una porta analoga che invece è l'uscita. Fin qui niente di speciale direte voi? Esatto fin qui niente di speciale, ma provate ora a gestire il puntino giallo con i tasti cursore e provate a farlo percorrere il sentiero giusto per la via dell'uscita. Provato? Si? Allora avete già capito che non dovete far toccare alcuna parete del labirinto a quel maledetto puntino. Se non riuscite a percorrere questo intricato groviglio di dedali potete sempre munirvi di un filo d'Arianna e riprovarci. Buon divertimento!!

Musica

Due sono le fantasie musicali di questo mese. Music è un breve programmino basic che trasformerà la tastiera del vostro Amiga in un potente sintetizzatore. Munitevi di carta e penna e iniziate a provare le varie combinazioni possibili, infatti non alleghiamo alcun commento al programma perché vogliamo lasciare al vostro genio musicale la straordinaria sensazione della composizione.

Interessante e molto più complesso si può definire invece Laboratorio del Suono, anche se in Basic questo programma offre apprezzevoli modi di gestione del suono su Amiga. Nella schermata principale troviamo un ricco menu che ci offre le varie possibilità di elaborazione del suono. Nella parte centrale dello schermo appare

la scritta relativa all'uso della stampante, per il momento fate finta di non vederla. A destra notate un riquadro con la scritta "USATE LE OPZIONI O PREMETE QUI", per Opzioni si intendono le possibilità di accendere o spegnere (ON-OFF) dei precisi strumenti di controllo del suono. Con l'opzione Attiv/Onda potete creare il tipo di onda sonora che più vi aggrada. Premendo il tasto del mouse quando il cursore si trova sull'ON di questa opzione si passa ad un altro schermo completamente dedicato alla costruzione della forma d'onda. Ora davanti a voi avete due rettangoli colorati sulla sinistra e delle istruzioni sulla destra. Per capire meglio come funziona questa parte del programma premete il tasto HELP. L'aiuto che troverete conterrà le seguenti note:

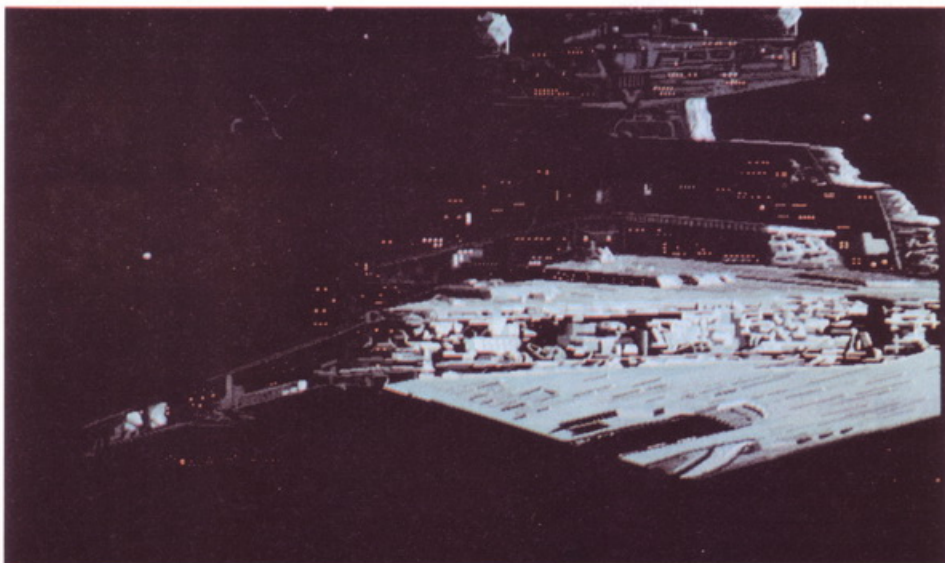
I due rettangoli colorati alla vostra sinistra sono un palette per il disegno della forma d'onda. L'onda si ottiene tenendo premuto il tasto sinistro del mouse e usando lo stesso per disegnarla. Se effettuate questo processo lentamente i punti verranno visualizzati più vicini l'uno all'altro e otterete così una rappresentazione di maggiore precisione. Ricordate che l'onda è disegnata di lato e che il rettangolo rosso ne è la parte inferiore. Questo programma è dotato di un dimensionatore automatico d'onda. Se l'onda da voi creata è inferiore ai 255 elementi potete ottenerne il dimensionamento fino al suddetto valore. Premendo F1 e scrivendo un numero tra -128 e 127 e premendo 'Return' otterrete il dimensionamento automatico. Quando avete creato la vostra forma d'onda potete ritornare alla schermata principale e scegliere l'opzione Ascesa/Discesa. I suoni creati useranno quindi l'onda da voi stabilita. Per ritornare alla forma di partenza spegnete (OFF) Attiv/Onda nello schermo principale. Ricordate che se decidete di generare piu' forme d'onda sulla stessa riga solo l'ultima sarà funzionante. Click mouse 2 volte per proseguire.

Passiamo dunque all'opzione Ascesa/Discesa. Selezionando l'ON passeremo ad una schermata interamente dedicata alle routine di frequenza del suono. Per prima cosa leggete attentamente i messaggi contenuti nella parte inferiore destra dello schermo. Anzi, per vostra comodità ve ne pubblichiamo i punti principali.

Premi 'ESC'per resettare a 0 le frequenze
Usate F1 per vedere il vostro sottoprogramma. Questa funzione và adoperata solo prima di provare il suono, ma dopo aver selezionato le frequenze
F2 riporta i valori del sottoprogramma a zero e lo cancella
F3 stampa il sottoprogramma. Siate ben sicuri di aver collegato la stampante e soprattutto che sia accesa
F4 Alterna la subroutine con un altra contenuta nel programma.

A questo punto premete due volte di seguito il tasto sinistro del mouse. Quattro indicatori di frequenza compariranno nella parte superiore dello schermo. Due per il primo canale e due per il secondo. Sulla destra potete notare le scritte:

FREQ 1 L
FREQ 1 H
FREQ 2 L
FREQ 2 H

Dove per 'H' (High) si intende alto e per 'L' (Low) basso. Potete spostare questi indicatori lungo le rispettive righe e nella parte inferiore sinistra veranno aggiornati numericamente i rispettivi valori di frequenza. Tenete presente che le basse frequenze (L) non possono essere uguali o superiori alle alte (H). Dopo aver posizionato i vari indicatori potete vedere la subroutine che li gestisce premendo F1. Per ascoltare le caratteristiche del suono premete il tasto del mouse su 'Ascolta suono'. Per ritornare alla schermata principale selezionate 'Menu P' (Menu Principale).

E ora vediamo assieme la schermata principale. Ricordate il messaggio relativò alla stampante? Bene, se l'avete, accendetela e clickate l'opzione stampa. Adesso premete il tasto sinistro del mouse posizionandovi nel riquadro 'USATE LE OPZIONI ...', nella parte alta dello schermo compare il messaggio:

SELEZIONATE I SUONI E QUINDI LA CASELLA RELATIVA IN BASSO

Nella parte alta sinistra dello schermo notate degli indicatori relativi al Suono, Frequenza, Tempo, Volume e Voce. Iniziate con il selezionare il numero di suoni desiderato e noterete che nella casella relativa a questi, nella parte inferiore del vostro monitor, ne viene riportato il valore numerico. Una volta deciso, clickate sulla scritta Suono nella relativa casella. La seconda selezione riguarda la Frequenza e si effettua nello stesso modo adoperato per i Suoni. Procedete fino alla Voce come appena illustrato. Se per caso volete riportare a zero il valore di qualcuna di queste componenti dovete premere il tasto sinistro del mouse essendovi pero' precedentemente posizionati sul rettangolino viola che si trova sotto a ciascuna casella. Vediamo assieme l'aiuto interno del programma.

Gran parte di questo programma può essere controllato per mezzo del mouse. Spostate gli indicatori tenendo premuto il tasto sinistro e quindi clickate la casella corrispondente per settare i valori. Per resettare a 0 i valori posizionatevi nei piccoli box viola sotto i vari Suono, Freq eccetera e premete il tasto del mouse. Le opzioni sulla destra vengono adoperate spostando il mouse e tenendone premuto il tasto sinistro.
Solo l'opzione MODO CORDA viene usata dallo schermo principale.

Sperando di essere stati sufficientemente esaurienti vi lasciamo alle vostre elucubrazioni sonore.

<<Ed egli intuonò una melodia orribile e tetra, che risuonava ai miei orecchi come un lugubre corno!
Oh cantore assassino, strumento della malvagità, innocente tra gli innocenti; già m'accingevo alla più bella tra le danze: quando tu coi tuoi suoni uccidesti il mio rapimento!>>.

F. NIETZSCHE

Grafica

Colori, colori, colori! Con Palette abbiamo voluto fornirvi un utile strumento di lavoro da usare da solo o come subroutine nei vostri programmi. Nell'ultimo caso basteranno alcune semplici modifiche. Il punto più interessante di Palette è quello di essere in grado di fornire il numero esadecimale del colore selezionato, in modo da permettervi l'uso diretto dei valori numerici nei vostri listati. Per uscire dal programma selezionate Cancel e premete il tasto sinistro del mouse.

BasicBoing è invece un programma in AmigaBasic che presenta un modo d'animazione un po' particolare ma sicuramente efficace. Per prima cosa il programma disegna un cubo e quindi calcola le differenti posizioni nelle quali farlo apparire. Una volta calcolate tutte le posizioni (quaranta), le immagini del cubo vengono scambiate velocemente l'una con l'altra, offrendo così un effetto animato.

Se ancora non siete soddisfatti e se vi considerate dei critici d'arte eccovi delle tele di gran pregio. Nella finestra dedicata alla grafica notate un icona con il nome SeeILBM. Questo programma serve alla visualizzazione dei disegni creati con De Luxe Paint in modo InterLace. Per visualizzare le immagini Astronave e Guerre_Stellari dovete dapprima selezionare l'icona SeeILBM, quindi portarvi sull'icona del disegno e premere due volte il tasto sinistro del mouse assieme allo SHIFT. Dopo aver ammirato le pregevoli tele di questa galleria potete ritornare al Workbench posizionando il cursore nell'angolo alto sinistro dello schermo e premendo il tasto del mouse.

Ai possessori del programma De Luxe Paint segnaliamo inoltre, che i suddetti disegni possono, ovviamente, essere caricati anche con quest'ultimo. E, dulcis in fundo, eccovi un regalo, nascosta chissà dove nel dischetto c'è un'avvenente signorina che sarà lieta di fare la vostra conoscenza.

<<Troppo m'addentrai nell'avvenire: fui colto da un brivido d'orrore.
E quando mi guardai intorno, vidi che il tempo era il mio solo contemporaneo.
Allora volai a ritroso, verso la patria, rapidamente: così venni a voi, esseri del presente, ...>>.
<<Ma come ciò m'accadde? Grande era la mia angoscia, e pur fui costretto a ridere! Non mai ancora il mio occhio aveva veduto cosa tanto variopinta e bizzarra!
Io rideva e rideva, mentre il mio piede ancor tremava e palpitava il mio cuore; "ma questa è la patria di tutti i vasi di colore" — dissi a me stesso>>.

F. NIETZCHE

Strumenti

Ed eccoci arrivati nel campo delle Utility che sicuramente vi aiuteranno nel vostro rapporto quotidiano con Amiga. DIRUTILITY è il programma che fa sicuramente al caso vostro. Quante volte vi siete ritrovati a riorganizzare il vostro archivio su disco e avete dovuto rinunciarvi, o perlomeno sospenderlo, a causa della lunga e noiosa scrittura di interminabili righe di comandi CLI?? Abbiamo pensato di farvi un favore nel proporvi questo comodo strumento di lavoro che vi faciliterà enormemente nella manipolazione dei dischetti.

Istruzioni

Prima di poter iniziare un qualsiasi processo, selezionate il file o la directory nella finestra che appare sulla sinistra dello schermo. Per prendere dimestichezza con il programma vi consigliamo di adoperare una copia di qualunque dischetto di lavoro. Per eseguire un qualsiasi comando dovete quindi clickare uno dei numerosi gadget che appaiono sulla destra. Nella parte inferiore della finestra trovate tre righe riservate alle varie stringhe di commento. Esse sono:

S: La directory corrente. Qui potete inserire un qualsiasi path (percorso) per arrivare al file o alla directory che vi interessa, ma normalmente non si usa in quanto tutte queste operazioni sono possibili usando il mouse nella finestra delle directory.

D: Il gadget di destinazione. Questo spazio è riservato per comunicare le directory di destinazione e viene adoperato anche per creare nuove directory.

Terza riga: Qui viene comunicato lo stato della finestra. Descrizione degli errori e relativo numero dell'errore nel DOS. Non provate ad inserire del testo in questa riga in quanto è impossibile, perlomeno in questo mondo.

DF0: DF1: DF2: RAM: HD0: —>
Setta la directory corrente al root della device selezionata.

ALL —>
Seleziona tutti i file della directory corrente, compresi quelli non visibili nella finestra.

CLEAR —>
Annulla la selezione di tutti i file preselezionati.

COPY —>
Copia i file selezionati nella directory visualizzata nella riga di commento D. Ricordate? Qui non potete inserire nomi di file ma solo path di directory già esistenti. Sarete in grado di inserire i path relativi alla directory corrente.

DELETE -->
Cancella i file selezionati. Non cancella le directory.

RENAME -->
Scambia il nome al primo file selezionato, o directory, con il nome o path (nella stessa device) che si trova nella riga D.

GETDIR -->
Se volete passare ad una nuova directory, selezionatela nella finestra sulla sinistra e quindi clickate questo gadget. Non usate questo comando per una directory che già è scritta nella prima riga di commento S:!! Se la dovete scrivere premete Return quando siete dentro la riga di commento, da quel momento quella sarà la nuova directory corrente.

MAKEDIR -->
Questo comando crea la directory descritta nella seconda riga di commento (D:), ovviamente potete usare i path necessari.

DELETEDIR -->
Questo comando cancella le directory vuote selezionate, se tentate di cancellare una directory con un qualche contenuto vi comparirà un messaggio di errore.

PARENT -->
Và alla parent directory.

ROOT -->
Si porta al root della device corrente.

TYPE -->
Potete usare questo gadget per leggere i vari file in formato ASCII. Selezionate i file che vi interessano e quindi clickate TYPE. Una nuova finestra viene aperta con il nome del file da voi scelto. Per avanzare di una pagina premete la barra spaziatrice, per far scrollare lo schermo di una sola riga premete Return mentre per portarvi alla fine del file usate il tasto ESC. Se avete selezionato più di un file il prossimo verrà automaticamente visualizzato. Dopo aver letto tutti i file prescelti questa finestra viene chiusa e si ritorna alla schermata iniziale.

SINFO -->
Viene usato per vedere il numero di byte disponibili del device appartenente alla directory corrente (S).

DINFO -->
Viene usato per vedere il numero di byte disponibili del device appartenente alla directory di destinazione (D).

PRINT -->
Funziona allo stesso modo di TYPE solo che l'output è indirizzato alla stampante.

SWAP -->
Si usa per scambiare il contenuto di S con D.

BYTE -->
Somma i byte dei file selezionati e visualizza il risultato ed il numero dei file.

EXEC -->
Esegue i file selezionati dall'interno di DirUtility. Se DirUtility è stato chiamato da CLI allora tutti gli output del programma saranno diretti al CLI richiamato.

SHOWILBM -->
Per rendere esecutivo questo gadget dovete possedere un qualsiasi programma in grado di visualizzare delle immagini, (vedi SeeILBM nella directory Grafica) e depositarlo nel dischetto che vi interessa, a questo punto DirUtility prova a copiare un comando di SHOW nella ram disk da SYS:C e quindi lo protegge, per evitare delle cancellazioni accidentali. A questo punto verrà dato l'execute per la visualizzazione dei file selezionati. Per passare da un'immagine all'altra basterà posizionarsi nell'angolo alto sinistro dello schermo e premere il tasto del mouse. Se SYS:C/SHOW non esiste il gadget SHOW rimane inutilizzabile.

NOTE AGGIUNTIVE
Se per caso avete selezionato tutti i file e quindi avete dato il DELETE e vi accorgete di aver commesso un'enorme sciocchezza, potete interrompere il processo clickando all'interno della finestra di visualizzazione della directory.

In omaggio vi offriamo una simpatica lente di ingrandimento che vi farà sentire alla stregua del famoso Sherlock Holmes. Caricando il programma 'Lente' potrete esaminare minuziosamente i dettagli più nascosti delle finestre della schermata Workbench. La finestra del programma può essere dimensionata a piacimento e sulla parte destra, ci sono dei gadget. Posizionatevi con il mouse sotto i gadget di sfogliamento delle finestre e premete ripetutamente il tasto sinistro del mouse e noterete che i dettagli dello schermo Workbench racchiusi nella finestra ingrandiscono ad ogni click. Per ritornare alla situazione iniziale posizionate il cursore un po' più su del gadget di dimensionamento e premete ripetutamente il tasto selezione del mouse. Certamente non è un programma di grande utilità ma può essere piacevole passare qualche minuto da segugi alla ricerca di indizi nelle varie finestre aperte con il Workbench. Se trovate qualcosa di interessante vi preghiamo di comunicarcelo al più presto.

Magazine

Il nostro viaggio si sta concludendo ma prima di lasciarci vogliamo spendere ancora qualche commento sulla parte Magazine. Quelli che già ci conoscono certamente saranno già andati a curiosare all'interno per trovare i programmi contenuti nella rivista. E sì?? È proprio così, in Magazine troverete tutti i listati sorgente riguardanti i vari articoli della rivista. Comunque, cari pigroni, siamo ben lieti di tenere tutta per noi la fatica della trascrizione, poiché vogliamo darvi il massimo e con il minimo sforzo, da parte vostra.

La separazione

Dopo questo viaggio all'interno di questa giungla di programmi ci sentiamo veramente stanchi, ma voi che invece avete ancora tutto il tempo per goderveli non esitate e spendete qualche ora con il vostro Amiga per fargli conoscere tutti i più nascosti meandri di questo dischetto, vedrete che saprà ricompensarvi adeguatamente.

Un celebre personaggio, non molto amato, diceva: " Chi si ferma è perduto! " Noi siamo ben distanti da quella persona ma ammettiamo che non siamo minimamente disposti a fermarci. Il numero tre di Amiga Magazine è già in cantiere, abbiate solo un po' di pazienza!!

<<E per cagion sua e dei suoi pari io devo dar compimento a me stesso: perciò ora fuggo la felicità e mi offro volentieri alla sventura — perché sia questa la mia ultima prova e l'ultima mia esperienza.

E in vero, era tempo che io me ne andassi; e l'ombra del viandante e il più lungo dei momenti e l'ora più silenziosa mi dicevan concordi: " è proprio tempo! ".

F. Nietzsche

CAPIRE E PILOTARE IL SUONO DI AMIGA

Ovvero il suono tra le dita

di Mr. Lambda

Disporre nel dispositivo audio di Amiga per capire come questo straordinario audio device lavori, e quindi sfruttare la capacità manipolatoria del Forth per creare

Ecco l'occasione per trasformare un incontro in un avvenimento determinante. E noi non vogliamo perderla. Come avrete già capito inizieremo subito Forth...issimo, ma siate sempre pronti al peggio. In ogni senso. Il Forth, infatti, è un linguaggio che offre, come pochi, ampie capacità di manipolazione e interattività, ma richiede pure, come pochi, spiccate doti di creatività. Qualsiasi cosa si voglia, bisogna essere innanzitutto capaci di progettarsela e realizzarsela da sè. Il programma che accompagna questo articolo è una valida dimostrazione di ciò. Insomma, la programmazione all'insegna dell'inventività e del bricolage. Ma credeteci, si può arrivare davvero molto lontano. E intanto incominceremo dal suono.

Il metodo della riproduzione del suono utilizzato da Amiga è simile a quello utilizzato dai Compact Disc che molti di voi ormai possiederanno. Anche se l'audio di Amiga non può essere paragonato con la fedeltà dei CD, la qualità del suono che può produrre risulta comunque impressionante. Ciò che limita le prestazioni di Amiga, se lo paragoniamo con il Compact Disc, è la frequenza di campionamento e il numero di bit a disposizione per campione. Infatti un disco digitale ha una risposta in frequenza che raggiunge i 20 kilohertz, mentre l'Amiga è limitato teori-

camente a circa 14 kilohertz, e praticamente consente una riproduzione corretta fino a 7,5 kilohertz cioè alla metà della cifra teorica precedente. Inoltre, come si diceva, un Compact Disc utilizza più bit per rappresentare ciascun punto della forma d'onda campionata e quindi ha un più ampio spettro dinamico.

Vediamo ora come l'Amiga produca effettivamente un suono. Innanzituto il suono che deve venire riprodotto deve essere definito come una serie di numeri che rappresentano il livello di voltaggio della forma d'onda campionata a determinati intervalli. I diversi voltaggi rappresentati da quei numeri devono quindi essere filtrati, amplificati, e inviati a uno o più altoparlanti per produrre finalmente il suono desiderato. In questa maniera è possibile creare qualsiasi suono concepibile e inconcepibile all'interno, naturalmente, dei limiti imposti dalla frequenza a cui la forma d'onda può venire campionata, che ne determinerà la risposta in frequenza, con il numero di bit disponibili per ciascun campione, che ne determinerà la quantità di rumore nel segnale e la dinamica, e, infine, con la memoria che abbiamo a disposizione.

Naturalmente per una più dettagliata comprensione della produzione del suono in Amiga vi suggeriamo di consultare la sezione riguardante l'audio nell'HARDWARE REFERENCE MANUAL.

Come tutto è iniziato...

Innanzitutto precisiamo che per la scrittura e il collaudo del programma che vi andiamo a presentare abbiamo utilizzato il Multi-Forth della Creative Solutions perché più facilmente reperibile e più completo. Le versioni più recenti di questo prodotto, tra le altre meraviglie, supportano anche un driver audio, rendendo quindi il tutto ancor più appetibile. Ma il Multi-Forth in nostro possesso, che è di gran lunga una versione più antica, per l'audio di Amiga non supporta alcunché. Ci siamo decisi a presentarvi questo programma innanzitutto perché rappresenta comunque un'ottimo esempio di applicazione del Forth alla soluzione di un problema fondamentale come quello del suono in Amiga e ne traccia in modo chiaro i passaggi e le possibilità permettendovi i più ampi sviluppi e miglioramenti, ma soprattutto perché non sarete certamente in molti a possedere le più recenti versioni del Multi-Forth e morirete, come noi, dalla voglia di poter controllare il suono di Amiga subito.

E veniamo ai fatti.

Dobbiamo incominciare con l'ammettere che realizzare il programma non è stato semplice e immediato come ci aspettavamo, ma eccovi il resoconto della nostra avventura.

Come ormai sapete, il nostro Multi-Forth non supportava alcunché per l'audio e non c'era alcun esempio nel manuale che ci potesse assistere. Insomma, possiamo dichiarare che non una parola del vocabolario a disposizione ci permetteva di fare alcunché con il suono! Ma questo è Forth! E se qualcosa vi è necessario, dovete costruirvela, diciamo bene?

Quasi subito fummo in grado di capire che avevamo bisogno di una funzione chiamata nei manuali BeginIO. I manuali indicavano anche che la funzione si aspettava come argomento l'indirizzo di una struttura IORequest. Ma nulla di tutto questo esisteva nel nostro Multi-Forth. E a nulla sono servite le ricerche dettagliate nel nostro manuale Multi-Forth: non c'era nulla, neanche qualcosa che potesse richiamare indirettamente questo concetto. La cosa che più ci ispirava era la funzione CallIO, ma questa non era chiaramente documentata, anzi non era documentata affatto, se non per una breve menzione nel glossario del manuale. Prima di incazzarci definitivamente con il nostro Multi-Forth, abbiamo considerato il fatto che non c'era alcuna documentazione sulla funzione BeginIO nemmeno negli AMIGA REFERENCE MANUAL. Il solo riferimento che avevamo potuto trovare era nella appendice B dello EXEC MANUAL. Esso consisteva semplicemente in questa macro in linguaggio assembler:

```
_BeginIO:
    move.l 4(sp),a1
    BEGINIO
    rts
```

Ma non ci è stata di molto aiuto.

Per farla breve abbiamo contattato tutti coloro che sembravano disposti a darci una mano e alla fine abbiamo deciso che il solo modo di risolvere questo problema era di arrangiarsi.

Ci siamo procurati una copia di Metascope — il debugger simbolico della Metadign — e un compilatore C, abbiamo compilato un programma demo che riguardava l'audio di Amiga, e quindi abbiamo incominciato a curiosare. Quasi immediatamente abbiamo individuato la famigarata funzione BeginIO e l'abbiamo disassemblata. Eccola:

```
_BeginIO:
    movea.l 4(a7),a1
    move.l  a6,-(a7)
    movea.l $14(a1),a6
    jsr     $-1E(a6)
    movea.l (a7)+,a6
    rts
```

Quando assegnamo come parametro l'indirizzo di una struttura IORequest, la funzione BeginIO dapprima preleva da quella struttura un puntatore a un device node, cioè ad un nodo del dispositivo software dell'audio, quindi utilizza quel puntatore come offset alla base di una jump table o tavola di salti nella memoria bassa e finalmente da là salta al ROM Kernel. Che cosa ci potrebbe essere di più semplice, eh?

Una delle difficoltà che si possono incontrare nella traduzione di routine in Forth risiede nel differente modo in cui C e Forth passano i parametri. Perché tutto funzioni a dovere in Multi-Forth, la prima istruzione deve infatti essere:

```
movea.l (a7)+,a1
```

che muove l'indirizzo della struttura IORequest dallo stack dati del Multi-Forth (A7 è lo stack pointer) nel registro A1. E ancora lo RTS deve essere sostituito da:

```
move.w (a2)+,d3
jmp    $18(a6,d3.w)
```

che è la macro per il NEXT in Multi-Forth.

Nel listato che accompagna questo articolo voi troverete un'altra definizione CODE per la funzione NewList. Naturalmente è possibile scrivere questa funzione anche direttamente in Forth, ma non ci siamo potuti trattenere dalla tentazione di continuare il lavoro iniziato con BeginIO! Per evitarvi la fatica di caricare l'assembler vi proponiamo una versione delle due word che inserisce i valori in codice macchina esadecimale direttamente nel vostro dizionario.

Ma a questo punto, era anche necessario implementare la chiamata OpenDevice della libreria Exec (che, per fortuna, è documentata nella nostra versione di Multi-Forth) per poter fornire a BeginIO un puntatore al device node. L'implementazione delle altre chiamate Exec è stata più semplice, e non si sono incontrati grossi problemi.

Un altro lavoro preparatorio riguardava la definizione delle strutture dati richieste dal device audio. Vediamone dapprima la versione in C e la versione in Assembler che ne danno i manuali e quindi la traduzione che ne abbiamo fatto in Forth. Vi facciamo presente che le recenti versioni del Multi-Forth includono tutte le strutture

disponibili presentate dai AMIGA REFERENCE MANUAL, ma la versione in nostro possesso invece non ci metteva a disposizione le strutture che intendiamo presentarvi e quindi ci costringeva ancora una volta a cimentarci con la nostra capacità di risolvere anche questo problema. Ma come abbiamo già avuto modo di vedere questo è proprio lo spirito del Forth.

Incominciamo con la struttura Message in C:

```
struct Message
{
  struct    Node mn_Node;
  struct    MsgPort *mn_ReplyPort;
  UWORD mn_Length;
}
```

E ora vediamone la versione in Assembler:

```
STRUCTUREMN,LN_SIZE
    APTR      MN_REPLYPORT
    UWORD   MN_LENGTH
    LABEL     MN_SIZE
```

E finalmente presentiamo la nostra versione in Forth:

```
structure Message
    LinkNode struct: +mnNode
                  ptr: +mnReplyPort
              short: +mnLength
structure.end
```

E ora interessiamoci della struttura IORequest, che, come potrete vedere, presenta delle particolarità. Innanzitutto presentiamo le due strutture utilizzabili in C:

```
struct  IORequest
{
  struct    Message io_Message;
  struct    Device *io_Device;
  struct    Unit *io_Unit;
  UWORD io_Command;
  UBYTE  io_Flags;
  BYTE    io_Error;
}
```

```
struct IOStdReq
{
  struct    Message io_Message;
  struct    Device *io_Device;
  struct    Unit    *io_Unit;
  UWORD io_Command;
  UBYTE  io_Flags;
  BYTE    io_Error;
  ULONG io_Actual;
  ULONG io_Length;
  APTR    io_Data;
  ULONG io_Offset;
}
```

E ora vediamo come queste strutture vengano tradotte in Assembler, considerando semplicemente la struttura IORequest un sottoinsieme della struttura IOStdReq:

```
STRUCTURE      IO,MN_SIZE
    APTR          IO_DEVICE
    APTR          IO_UNIT
    UWORD       IO_COMMAND
    UBYTE        IO_FLAGS
    BYTE          IO_ERROR
    LABEL        IO_SIZE
    ULONG       IO_ACTUAL
    ULONG       IO_LENGTH
    APTR          IO_DATA
    ULONG       IO_OFFSET
    LABEL        IOSTD_SIZE
```

Per adeguare queste strutture al Forth si adotta il medesimo concetto visto nella precedente traduzione Assembler anche se la sintassi chiaramente ne differisce:

```
structure IOStdReq
  structure IORequest
      Message struct: +ioMessage
                          ptr: +ioDevice
                          ptr: +ioUnit
                      short: +ioCommand
                        byte: +ioFlags
                        byte: +ioError
structure.end          IORequest +
                        long: +ioActual
                        long: +ioLength
                          ptr: +ioData
                        long: +ioOffset
structure.end
```

Consideriamo brevemente il costituirsi di questa struttura e la soluzione programmativa che le permette di operare correttamente. La word STRUCTURE crea una nuova word, fondamentalmente una costante, e lascia l'indirizzo del campo parametri (parameter field) di quella word e uno zero nello stack. Lo zero è un valore fittizio in cui viene accumulata la dimensione della struttura. Quando STRUCTURE.END viene incontrata la dimensione accumulata della struttura viene memorizzata all'indirizzo del campo parametri della word. Come risultato di queste operazioni, quando viene eseguito il nome della struttura, ne viene restituita la dimensione. Dal momento che il nome della struttura ritorna la sua dimensione, l'inserzione di ' IORequest + ' alla fine della struttura IORequest aggiunge l'offset appropriato perché +ioActual venga eseguita correttamente.

E finalmente consideriamo il template della struttura dati utilizzata dal device audio (audio.device). Essa include una struttura IORequest seguita da informazioni riguardanti la locazione e la lunghezza della tavola dati del suono, il periodo e il volume della forma d'onda e il numero di volte che il suono deve essere ripetuto. Incominciamo considerandone la struttura in C:

```
struct Audio
{
  struct     IORequest ioa_Request;
  WORD     ioa_AllocKey;
  UBYTE    *ioa_Data;
  ULONG   ioa_Length;
  UWORD  ioa_Period;
  UWORD  ioa_Volume;
  UWORD  ioa_Cycles;
  struct     Message ioa_WriteMsg;
}
```

Ecco la versione in sintassi Assembler:

```
STRUCTURE Audio,IO_SIZE
WORD        ioa_AllocKey
APTR         ioa_Data
ULONG      ioa_Length
UWORD     ioa_Period
UWORD     ioa_Volume
UWORD     ioa_Cycles
STRUCT     ioa_WriteMsg,MN_SIZE
LABEL       ioa_SIZEOF
```

E quella in Multi-Forth:

```
structure IOAudio
    IORequest struct: +ioaRequest
                       short: +ioaAllocKey
                           ptr: +ioaData
                        long: +ioaLength
                       short: +ioaPeriod
                       short: +ioaVolume
                       short: +ioaCycles
      Message struct: +ioaWriteMsg
structure.end
```

Tutte le costanti che sono comandi del device e che trovate nel listato sono spiegate dettagliatamente negli AMIGA REFERENCE MANUAL.

Vi abbiamo messo a disposizione, traducendole in sintassi Forth, le funzioni CreatePort() e DeletePort() che sono disponibili in C e che si rivelerranno molto utili. CreatePort richiede e inizializza una message port che vi mette a disposizione un meccanismo per la comunicazione tra differenti task e interrupt. Una porta puo' essere sia pubblica, cioè con un nome e conosciuta al resto del sistema sistema:

```
0" nome" priorità CreatePort
```

Oppure privata, e cioè senza nome e non conosciuta al resto del sistema:

```
0 priorità CreatePort
```

Nel caso dell'audio device noi abbiamo utilizzato una porta privata e senza nome dal momento che nessun altro task nel sistema aveva necessità di sapere della sua esistenza.

Noi ci scusiamo fin d'ora se non riusci-

remo a soddisfare ogni vostro dubbio, ma il metodo migliore per comprendere a fondo questo nostro programma è studiarlo consultando opportunamente gli AMIGA REFERENCE MANUAL. E non trascurate lo EXEC REFERENCE MANUAL anche se all'inizio vi può sembrare ostico e difficile de comprendere. Noi abbiamo trovato il manuale veramente chiaro sotto molti punti di vista, anche se naturalmente necessita di essere letto molte volte per riuscire ad afferrare alcuni punti più complessi. A ogni successiva lettura ci siamo accorti che ogni cosa ci riusciva più chiara e conseguente.

Ed è diventato suono

Il listato che vi presentiamo vi permette di generare semplici forme d'onda all'interno di un limitato ambito di frequenze. Vi forniamo le tavole dei dati per l'onda sinusoidale (Sine), l'onda quadra (Square), l'onda triangolare (Triangle) e l'onda a dente di sega (Sawtooth) che possono produrre buoni risultati tra i 1200 e i 2400 Hz. Anche frequenze tra 600 e 1200 Hz possono essere riprodotte con buoni risultati anche se ciò non è strettamente in accordo con quanto enunciato nel manuale dello Hardware. Le frequenze al di fuori di questi limiti produrranno suoni distintamente distorti.

Ciascuna tavola di dati audio consiste di dodici campioni che descrivono un ciclo completo. Le frequenze più basse richiederebbero una tavola di dati più lunga, e le frequenze più alte meno campioni. Per creare un suono continuo basta che puntiate il DMA audio alla tavola dei dati dell'onda e gli comunichiate quante volte desiderate ripeta i dati. Un ciclo è sufficiente per generare un suono continuo. Il DMA si riposiziona automaticamente all'inizio dell'array dei dati ogni volta che raggiunge la sua fine e ripete questa operazione fino a quando ha riletto l'array per il numero di volte che avete specificato.

E adesso vi diamo alcune spiegazioni che riguardano la word Hz. Il manuale dello Hardware indica un metodo per determinare adeguatamente la frequenza di campionamento necessaria a produrre una data frequenza da un precostituito numero di campioni che noi chiaramente non abbiamo seguito. Questo metodo è basato sul periodo del timer audio e il periodo della frequenza desiderata, ma è molto più semplice utilizzare le frequenze per eseguire i calcoli necessari in modo da evitare reciproci o calcoli con virgola mobile. Ecco l'equazione:

Frequenza_di_Campionamento = AudioClock / (Frequenza_Desiderata * Campioni)

Dove Frequenza_di_Campionamento è il numero di tick che il clock dell'audio deve aspettare prima di prelevare un campione, la frequenza di AudioClock è 3,579546 Mhz, e Campioni è il numero di campioni che devono essere riprodotti in un periodo della forma d'onda.

La word Hz calcola il periodo richiesto per generare una determinata frequenza da una tavola di dodici campioni e memorizza quel valore in una variabile chiamata Period per servirsene successivamente.

Ora parleremo delle altre word descrivendo l'azione della word Wave. La prima cosa che Wave fa è tentare di aprire una port audio. La word AudioPort? ci restituisce l'indirizzo della port allocata dalla word CreatePort, se, naturalmente questa ha successo, e il flag 0 se questa non ha successo. Se il tentativo di creare una message port non ha successo Wave termina con il messaggio "La port del device audio non è stata aperta."

Se la message port è stata aperta con successo Wave continua inizializzando dapprima la struttura ioaRequest della struttura IOAudio sound con i valori richiesti dalla chiamata di sistema OpenDevice.

```
\
\
\  AudioDemo
\
\

anew AudioDemo

\ Ecco il codice Assembler di cui
\ abbiamo bisogno per includere
\ le funzioni BeginIO e NewList
\ nei nostri programmi.

\ CODE BeginIO ( IORequest -- )
\  SP )+ A1 LONG MOVEA,
\   A6 A7 -) LONG MOVE,
\   20 A1 I) A6 LONG MOVEA,
\   -30 A6 I) JSR,
\   A7 )+ A6 LONG MOVEA,
\  NEXT END-CODE
```

```
HEX  \  e per evitarvi la fatica
     \  di caricare l'Assembler,
     \  eccovi una versione
     \  alternativa.

CREATE BeginIO ( IORequest -- )
  -4 ALLOT 225F w, 2F0E w,
   2C69 w, 0014 w, 4EAE w,
   FFE2 w, 2C5F w, 361A w,
   4EF6 w, 3018 w,

\ CODE NewList ( list -- )
\   SP )+ A0 LONG MOVEA,
\   A0 A0 () LONG MOVE,
\   A0 () 04 LONG ADDQ,
\   4 A0 I) LONG CLR,
\   A0 8 A0 I) LONG MOVE,
\   NEXT END-CODE

CREATE NewList ( list -- )
  -4 ALLOT 205F w, 2088 w,
```

```
   5890 w, 42A8 w, 0004 w,
   2148 w, 0008 w, 361A w,
   4EF6 w, 3018 w,

DECIMAL

: AllocMem
( dimensione_in_byte\caratteristiche
  -- blocco_di_memoria )
  !d1 !d0 exec@ 33 ;

: FreeMem
( blocco_di_memoria\dimensione_in_byte
  -- )
  !d0 !a1 exec 35 ;

: AllocSignal
( num_segnale -- num_segnale )
  !d0 exec@ 55 ;

: FreeSignal ( num_segnale -- )
  !d0 exec 56 ;

: AddPort ( port -- )
  !a1 exec 59 ;

: RemPort ( port -- )
  !a1 exec 60 ;

: OpenDevice
( nome_dev\unita'\ioRequest\flag
  -- errore )
  !d1 !a1 !d0 !a0 exec@ 74 ;
```

```
: CloseDevice ( ioRequest -- )
  !a1 exec 75 ;

: WaitIO ( ioRequest -- error )
  !a1 exec@ 79 ;

SYMTABLE DEFINITIONS

4 wconstant NT_MSGPORT

0 wconstant PA_SIGNAL

structure Message
  LinkNode struct: +mnNode
                 ptr: +mnReplyPort
               short: +mnLength
structure.end

1 0  scale wconstant MEMF_PUBLIC
1 16 scale constant  MEMF_CLEAR


\ osservate bene ora un esempio
\ di struttura nidificata

structure IOStdReq
  structure IORequest
    Message struct: +ioMessage
                  ptr: +ioDevice
                  ptr: +ioUnit
               short: +ioCommand
                byte: +ioFlags
                byte: +ioError
```

```
  structure.end  IORequest +
                 long: +ioActual
                 long: +ioLength
                  ptr: +ioData
                 long: +ioOffset
structure.end

1 wconstant IOF_QUICK
3 wconstant CMD_WRITE
0" audio.device" constant AUDIONAME
1 4 scale wconstant ADIOF PERVOL
structure IOAudio
  IORequest struct: +ioaRequest
               short: +ioaAllocKey
                 ptr: +ioaData
                long: +ioaLength
               short: +ioaPeriod
               short: +ioaVolume
               short: +ioaCycles
     Message struct: +ioaWriteMsg
structure.end

FORTH DEFINITIONS

\ Il formato adeguato per creare
\ una port senza nome e quindi
\ privata e':
\ 0 pri CreatePort

: CreatePort ( 0" nome"\pri --
                  MsgPort oppure 0 )
        LOCALS| pri name |
  -1 AllocSignal

\ controlliamo innanzitutto se
\ abbiamo ottenuto un bit per il
\ segnale

dup -1 = not

\ e' gia' allocata la memoria per la
\ nostra port ? Se no, facciamolo

IF MessagePort
   MEMF_CLEAR MEMF_PUBLIC OR
   AllocMem dup 0=

\ abituale controllo sul risultato
\ dell' allocazione
\ in caso di risultato negativo
\ concludiamo inserendo nello stack
\ uno zero o NULL

     IF DROP FreeSignal 0

\ altrimenti ci decidiamo ad
\ inizializzare la struttura
\ MessagePort

     ELSE
       name         over +mpLinkNode
                              +lnName !
       pri          over +mpLinkNode
                              +lnPPri C!
       NT_MSGPORT over +mpLinkNode
                              +lnType C!
       PA_SIGNAL    over +mpFlags C!
       swap         over +mpSigBit C!
       0 FindTask over +mpSigTask !
       dup name

\ se la port e' 'pubblica', cioe'
\ ha un nome, AddPort permette al
\ resto del sistema di individuarla
\ e 0" name" FindPort ci restituira'
\ il suo indirizzo

    IF AddPort
    ELSE +mpLinkHeader NewList THEN
      THEN
    ELSE drop 0
    THEN ;

: DeletePort  ( port -- )
    LOCALS| port |
    port +mpLinkNode +lnName @
    IF port RemPort
      THEN
    255 port +mpLinkNode +lnType C!
    -1  port +mpLinkHeader +lhHead !
    port +mpSigBit C@ FreeSignal
    port MessagePort FreeMem ;

struct IOAudio sound
sound IOAudio ERASE

DECIMAL

CREATE ChannelMask 15 C,
\ in questo modo potremo utilizzare
\ tutti e quattro i canali
\ disponibili
```

```
CREATE Samples 12 ,
\ la lunghezza dei dati del suono

CREATE Period 400 W,
\ intervallo che intercorre tra
\ un campione e il successivo
\ in tick del timer

CREATE Duration 400 W,
\ numero di volte che vogliamo far
\ ripetere il suono

VARIABLE >Sound
\ puntatore ai dati del suono

: :AudioData  ( nome ( -- )
  CREATE DOES> >Sound ! ;

  :AudioData Sine
    0 c,  63 c,  109 c,
     126 c,  109 c,  63 c,
    0 c, -63 c, -109 c,
    -126 c, -109 c, -63 c,

  Sine  \ forma d'onda di default

  :AudioData Square
    80 c,  80 c,  80 c,
    80 c,  80 c,  80 c,
   -80 c, -80 c, -80 c,
   -80 c, -80 c, -80 c,

  :AudioData Triangle
    0 c,  42 c,  84 c,
  126 c,  84 c,  42 c,
    0 c, -42 c, -84 c,
 -126 c, -84 c, -42 c,

  :AudioData Sawtooth
    0 c,   25 c,  50 c,
   57 c,  100 c, 125 c,
 -125 c, -100 c, -75 c,
  -50 c,  -25 c,   0 c,

3579546 ( Hz ) CONSTANT AudioClock
\ intervallo di frequenza del timer

: Hz ( freq -- )
  Samples @ * AudioClock swap /
  Period W! ;
```

```
: FreeAudioPort ( -- )
   sound +ioaRequest +ioMessage
   +mnReplyPort @ DeletePort ;

: AudioPort? ( -- MsgPort oppure 0 )
   0 0 CreatePort dup
   IF dup sound +ioaRequest
   +ioMessage +mnReplyPort !
   THEN ;

: AudioDevice? ( -- vero oppure falso )
   AUDIONAME 0 sound 0 OpenDevice NOT ;

: initSoundPort  ( -- )
   10 sound +ioaRequest +ioMessage
   +mnNode +lnPpri C!
   ChannelMask sound +ioaData !
   1 sound +ioaLength ! ;

: initSoundStruct ( -- )
   CMD_WRITE sound +ioaRequest
   +ioCommand W!
   ADIOF_PERVOL IOF_QUICK or sound
   +ioaRequest +ioFlags C!
   >Sound     @ sound +ioaData    !
   Duration W@ sound +ioaCycles W!
   Samples    @ sound +ioaLength  !
   Period   W@ sound +ioaPeriod W!
   64          sound +ioaVolume W!
   ;

: Wave ( -- )
   AudioPort?
   IF initSoundPort
     AudioDevice?
     IF initSoundStruct
  sound BeginIO sound WaitIO
  drop sound CloseDevice
ELSE
." Il device audio non e' stato aperto."
cr
     THEN FreeAudioPort
   ELSE
." La port audio non e' stata aperta."
   THEN
;

\ ed eccoci, finalmente, con un esempio
\ all'utilizzo del programma:
\   300 Hz Sine Wave
```

Disegno realizzato con Amiga 2000 e stampato con Xerox 4020.

La priorità richiesta viene memorizzata nel campo +lnPpri della struttura IOAudio. A proposito di +lnPpri permetteteci un commento. Probabilmente a molti di voi sarà sembrato un errore o un refuso, dal momento che si riferisce a LN__PRI della relativa struttura in Assembler oppure a ln__Pri della relativa struttura in C, e invece è proprio così. Ma tant'è. E dal momento che così appare nella nostra versione di Multi-Forth, noi siamo stati costretti ad accettarlo proprio così, nudo e crudo, e sbagliato. Quindi per non perdere il controllo del vostro sistema nervoso per un errore che vi sembra sciocco e impossibile, date un'occhiata alla word relativa che è presente nella vostra versione. Bene. E ora continuiamo. Si diceva della word Wave e delle sue successive inizializzazioni. Dopo la priorità, l'indirizzo di ChannelMask viene memorizzato nel campo +ioData, che richiederà per noi al sistema tutti quattro i canali. La mask per la richiesta dei canali consiste in un solo byte, e così +ioaLength viene settato a uno.

La word AudioDevice? tenta poi di aprire il device audio con una chiamata alla funzione della libreria EXEC OpenDevice e ritorna un flag vero se ha successo. Se la chiamata ha successo la struttura IOAudio viene riinizializzata affinché punti ai dati per la generazione della forma d'onda. Altrimenti l'esecuzione termina con il messaggio "Il device audio non è stato aperto." e quindi viene liberata la memoria allocata per la message port. Se l'apertura del device audio ha successo e la funzione OpenDevice ha memorizzato l'indirizzo del device node nel campo +ioDevice, l'indirizzo della struttura sound viene passato alla funzione BeginIO che abilita il DMA audio e produce il suono. WaitIO attende che il suono finisca, e CloseDevice abbandona il device audio. FreeAudioPort restituisce allo heap del sistema la memoria allocata per la message port.

La sintassi appropriata per generare un suono è:

```
1200 Hz Sine Wave
2000 Hz Square Wave
```

e così via. Naturalmente, una volta che la frequenza sia stata specificata utilizzando Hz, non è più necessario ripetere il comando se intendiamo generare un altro tipo di forma d'onda della stessa frequenza. In altre parole, una volta che il comando 2000 Hz viene eseguito, Triangle Wave oppure Sawtooth Wave genereranno le forme d'onda appropriate alla frequenza 2 kHz. E ancora, dopo che siano stati specificati la frequenza e i dati della forma d'onda, Wave può venire utilizzato nuovamente per rigenerare lo stesso tono.

Bene, ora che la strada verso il controllo del suono in Forth è stata aperta non possiamo che augurarci che possiate proseguire nel migliore dei modi. Al prossimo incontro.

Original Software by C.T.O.

Codice	Titolo	Prezzo di vendita
	Manuali in Italiano	
	ACTIVISION	
ACT101	Hacker II	29.500
ACT103	Shanghai	29.500
ACT104	GBA Championship Golf	29.500
ACT105	GBA Championship Basketball	29.500
ACT106	Championship Baseball	29.500
ACT107	GFL Championship Football	29.500
ACT108	Borrowed Time	33.000
ACT109	Little Computer People	33.000
ACT110	Mindshadow	33.000
ACT111	Tass Times	33.000
ACT112	Portal	48.000
ACT002	Space Quest	35.000 in inglese
	S.P.A.	
SPA101	The Art of Chess	29.500
	ELECTRONIC ARTS	
ECA101	Adventure Construction Set	38.000
ECA102	Artic Fox	29.500
ECA103	Bard's Tale I	29.500
ECA104	Chess Master 2000	29.500
ECA105	Earl Weaver Baseball	29.500
ECA106	Instant Music	33.000
ECA107	Marble Madness	29.500
ECA108	Skyfox	29.500
ECA109	Test Drive	33.000
ECA110	Ferrari Formula One	38.000
ECA601	Art Part I	34.000
ECA602	Art Part II	34.000
ECA603	Hot & Coll Jazz	34.000
ECA604	Rock 'n' Roll	34.000
ECA605	Seasons & Holidays	34.000
ECA701	DELUXE Music Construction Set	94.000
ECA702	DELUXE Paint II	99.000
ECA703	DELUXE Print	90.000
ECA704	DELUXE Video	109.000
ECA011	Black Shadow	33.000 in inglese
	THE DISC COMPANY	
TDC001	Kind Words	60.000
	PROGRESSIVE PERIPHERALS & SOFTWARE	
PPS001	PIXmate	94.000
	TYNESOFT	
TYN001	Winter Olympiad 88	27.000 in inglese
	COMMODORE ITALIANA S.p.A.	
CIT001	Pagesetter	210.000
CIT002	Texteraft Plus	145.000 in inglese
	C.T.O. s.r.l.	
LGX002	Logistix	120.000
SBA004	Superbase Personal	190.000
SBA005	Superbase Professional	399.000 in inglese

BUONO D'ORDINE

Desidero ricevere i seguenti articoli:

CODICE	TITOLO GIOCO	PREZZO
	TOTALE L.	

Cognome ....

Nome ....

Via ....

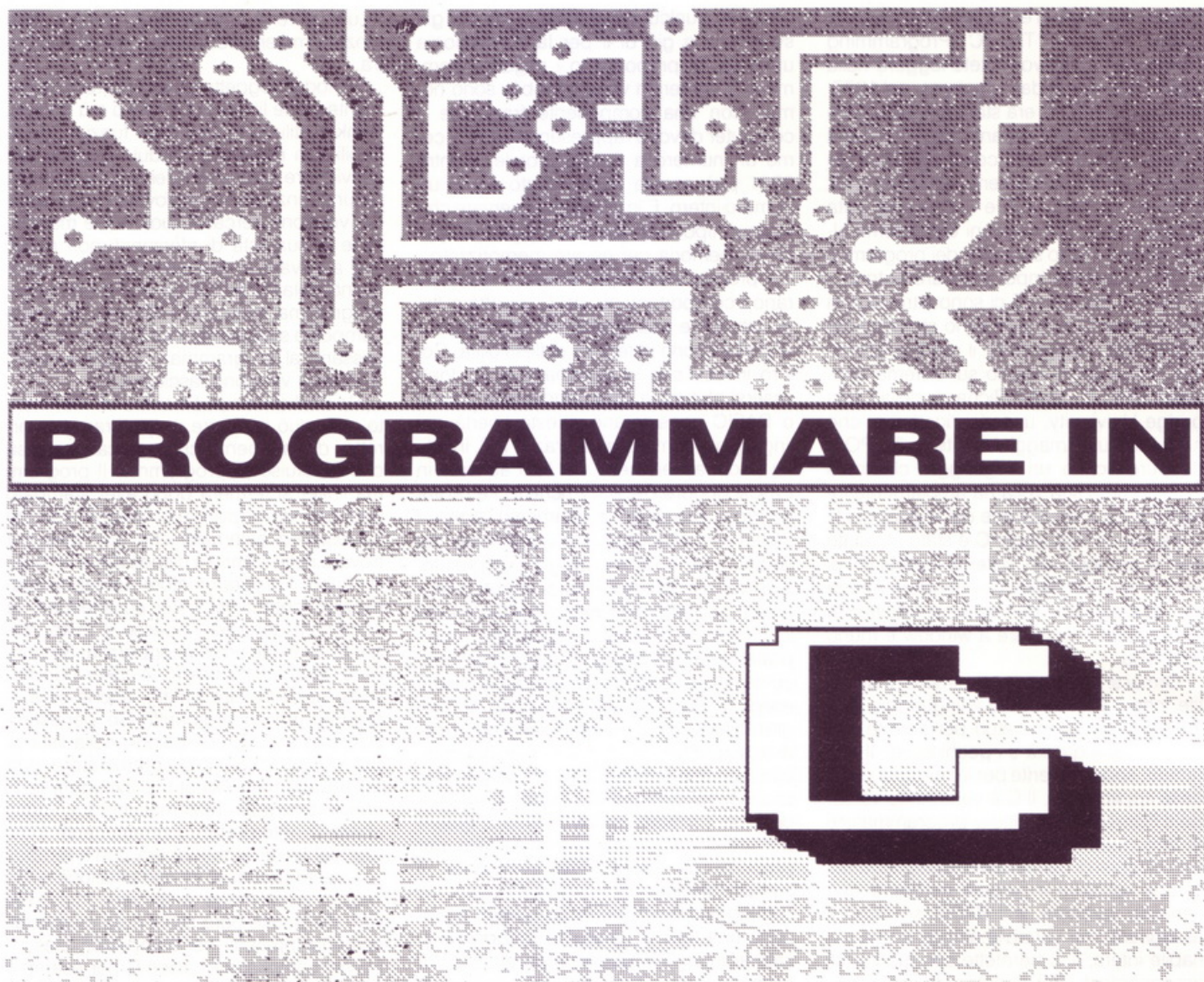
Cap. .... Città ....

Prov. .... Tel. ....

Firma ....

(se minorenne quella di un genitore)
Gli ordini non firmati non verranno evasi.
Completa le parti del buono d'ordine (o di una sua fotocopia) e spediscilo in busta chiusa a:
AMIGA SOFT SERVICE - Via Rosellini, 12 - 20124 Milano

AMIGA SOFT SERVICE

Per pagamento in contrassegno addebito di L. 6.000 per importi inferiori a L. 35.000; nessun addebito per ordini superiori a L. 35.000.
Pagamento con assegno da allegarsi al coupon d'ordine intestato a: "LENA s.r.l." addebito forfettario di L. 4.000 per importi inferiori a L. 30.000; nessun addebito per ordini superiori a L. 30.000.
Le spedizioni verranno effettuate da "LENA s.r.l." - I prezzi sono intesi al pubblico I.V.A. inclusa.

Patti chiari, amicizia lunga

di Mr. Lambda

Avviarsi sulla luminosa strada della programmazione con il linguaggio C è possibile a patto che sappiamo adattare le nostre conoscenze di programmazione al nuovo ambiente oppure ci accostiamo fin dall'inizio con idee chiare e precise alle possibilità offerte da questo sorprendente linguaggio. Ed è proprio questo lo scopo di questo nostro incontro. Quindi inforcate i vostri occhiali da sole e seguiteci.

Negli articoli che via via vi introdurranno al linguaggio, il C non sarà lasciato a se stesso, ma volta per volta verrà messo in relazione ad altri linguaggi come il BASIC, il Pascal, l'Assembly, ecc. E questo perché riteniamo che alcuni aspetti del C saranno facilmente appresi da coloro che conoscano sufficientemente un altro linguaggio di programmazione. Prendiamo, per esempio, la funzione C chiamata printf() che stampa messaggi sullo schermo. Il suo nome significa ' print formattato '. Se la parola ' funzione ' vi e' nuova, ma non lo dovrebbe essere, potete sostituirla con le parole ' subroutine ' oppure ' procedura '. Una funzione e' una parte di una programma che esegue un determinato lavoro, come stampare messaggi sullo schermo, oppure leggere file dal disco.

Ma prima di continuare concediamoci alcuni brevi riferimenti storici. La maggior parte delle Introduzioni al linguaggio C comprendono riferimenti storici per parecchie buone ragioni. Innanzitutto calarsi nella giusta atmosfera e quindi comprenderne i rapporti con l'evoluzione informatica.

Brian Kernighan e Dennis Ritchie hanno scritto il libro "The C Programming Language", che voi potete leggere nella traduzione offerta dalla Jackson, nel 1977. Ma il linguaggio C era stato già sviluppato agli inizi degli anni settanta sotto il sistema operativo Unix. Infatti, come già dovreste sapere, a tutt'oggi il sistema operativo Unix è scritto proprio in C e il C a sua volta adotta alcune convenzioni del sistema Unix, come il modo standard dei programmi in C di accettare input e inviare output.

Ma c'è dell'altro. E ci sono altre ragioni per questo. C e Amiga sono strettamente imparentati. AmigaDOS, il sistema operativo di Amiga, è basato sul sistema operativo TRIPOS. Martin Richards della Cambridge University, uno dei ricercatori che ha contribuito maggiormente a TRIPOS, ha pure scritto un linguaggio chiamato BCPL, che a sua volta ha ampiamente influenzato la formulazione del linguaggio C e insieme al linguaggio C è stato utilizzato per lo sviluppo del sistema operativo di Amiga. Ma non basta. Commodore-Amiga ha utilizzato il C per scrivere Intuition e Workbench, il software a window e menu di Amiga.

Ormai si può dire, senza timore di smentite, che il C è divenuto il linguaggio preferito tra i programmatori per tutte le sue straordinarie qualità e i pochi difetti. Il C è rapido, soddisfacente per la maggior parte delle applicazioni. Il C è veramente versatile e flessibile. Permette al programmatore di fare quasi ogni cosa. Mentre il Pascal può rifiutarsi di permettere l'assegnamento di una variabile in virgola mobile a una variabile carattere, e il BASIC si rifiuta di assegnare una variabile stringa a una variabile intera, il C permette con facilità tali assegnamenti. Per quanto siano possibili, queste mescolanze di tipi di dati non sono consigliabili, ma in ogni caso il C assume che voi sappiate che cosa state facendo e compie esattamente ciò che gli avete chiesto.

Questa caratteristica del linguaggio viene chiamata ' typing '. IL Pascal e' un linguaggio fortemente tipizzato, dal momento che l'operatore ' := ' non ci consente di assegnare un numero in virgola mobile a una variabile carattere. Il BASIC e' meno fortemente tipizzato , dal momento che ' = ' assegnera' numeri in virgola mobile ad interi e viceversa. Tuttavia il C è debolmente tipizzato, anche se offre una varietà e complessità di tipi di dati di tutto rispetto. Il compilatore può produrre un avvertimento o warning nel caso che i tipi di dati non corrispondano, ma continuerà a fare il suo dovere fino in fondo e il programma girerà ugualmente.

Ma se qualcuno a questo punto sta già svenendo o giù di lì per la terminologia utilizzata, vi proponiamo i seguenti commenti. I numeri in virgola mobile sono numeri con una componente frazionale. I computer devono rappresentare 7,302 come un numero a virgola mobile, mentre 3751 può essere memorizzato come un numero intero. È importante osservare che ciascun tipo una quantità diversa di spazio di memoria per essere memorizzato.

I programmatori più smaliziati considerano il comportamento del C rispetto ai tipi di dati come una qualità del linguaggio e vi si riferiscono in termini di flessibilità. Coloro invece che si avvicinano al C dopo essersi abituati a programmare in Pascal o BASIC considerano questa libertà del linguaggio come una disgrazia. Il C, infatti, ha pochi avvertimenti del tipo abituale in Pascal oppure BASIC. Anche il C è un linguaggio compilato esattamente come il Pascal o il Modula-2 e diversamente dal BASIC, che è un linguaggio interpretato. E se voi conoscete il Pascal oppure il Modula-2 apprenderete molto più facilmente il C. Quando si parla di linguaggio ' compilato ' significa che un programma viene convertito in un'altra forma prima di essere eseguito. Per i linguaggi compilati, un programmatore crea un testo sorgente o codice sorgente con un editor, e quindi compila il codice sorgente con il compilatore, producendo un codice oggetto o modulo oggetto. I moduli oggetto devono essere poi linkati con un programma linker per produrre un programma eseguibile. Il compilatore traduce il codice sorgente nel linguaggio macchina del computer, e produce una lista di variabili e funzioni utilizzate all'interno di questo programma. Per esempio, un programma puo' utilizzare la funzione printf(). Questa funzione può far parte del linguaggio, ma deve essere definita da qualche parte al di fuori dal codice sorgente, a meno che non la ridefiniate voi per l'occasione. La funzione printf() viene detta appunto esterna o non risolta, dal momento che il compilatore non può generare codice macchina per essa. Un modulo oggetto è composto da questa mescolanza di linguaggio macchina e liste di riferimenti non risolti (unresolved references).

Dopo che il programma è stato compilato, tutti questi riferimenti alle funzioni e alle variabili devono essere risolti con il linker.

Se voi comprerete un compilatore C, insieme con esso avrete a disposizione diverse collezioni di funzioni pre-scritte e pre-compilate, chiamate librarie. Le funzioni standard come printf() sono incluse in una libreria, mentre sin(), cos() e altre funzioni matematiche sono incluse in un'altra.

Il modulo oggetto prodotto precedentemente viene linkato con una o più librerie. Il linker utilizza la lista delle funzioni e variabili non risolte del modulo oggetto per individuare ciascuna libreria di cui servirsi. Se una funzione viene trovata in una libreria, vengono cercati il suo codice macchina e la sua lista di riferimenti non risolti, fino a trovare e risolvere tutti i riferimenti. Quindi solamente il linker può produrre un programma eseguibile. Questo file viene composto solamente dalle funzioni che necessarie al programma. Le altre funzioni di libreria vengono lasciate fuori.

Se il linker non può trovare un riferimento, non può produrre un programma corretto, dal momento che manca qualcosa che costituisce il programma. Il programma linker si fermerà dopo aver stampato una lista di funzioni e/o variabili esterne (unresolved externals) che non riesce a trovare nelle librerie a disposizione. In questo caso, il programmatore dovrebbe esaminare l'output del linker ed effettuare le correzioni del caso. Spesso il testo sorgente contiene errori di ortografia. Se printf() è stato scritto prnitf(), il linker non riuscirà mai a trovare prnitf(), anche se è in grado di individuare printf(). Se, per esempio, sin() e cos() compaiono nella lista di output del linker, il programmatore può anche aver dimenticato di includere la libreria di funzioni matematiche. La vostra abilità di interpretare correttamente l'output del linker si svilupperà con il tempo.

Questo processo di compilazione e linkaggio vi costringe a scrivere molto nel CLI. Voi potete però servirvi sia di speciali comandi presenti nei compilatori stessi - LC nel compilatore Lattice C per esempio -, sia di particolari file batch o execute chiamati MAKE. Nell'articolo dedicato al C del numero precedente della rivista vi abbiamo appunto presentato un'utility MAKE che vi permette di utilizzare efficacemente il compilatore Lattice C 3.10. Se voi avete creato un programma C chiamato Prova.c e volete compilarlo, è sufficiente che scriviate sulla linea comandi del CLI.

```
EXECUTE MAKE PROVA
```

E il gioco è fatto. Come vi abbiamo già spiegato il file batch o execute contiene tutti i comandi CLI necessari per la compilazione e il linkaggio di un programma C, evitandovi un mucchio di digitazione ad alto rischio!

Primi passi

Esaminiamo con attenzione un breve e semplice programma in C, BASIC, e Pa-

scal. Il programma moltiplicherà due numeri e stamperà il risultato. Eccolo in C:

```
#include "stdio.h"

/* un programma per moltiplicare due numeri */

main()
{
int a,b,c;

  a = 3;
  b = 4;
  c = a * b;
  printf("Il risultato è %d\n",c);
}
```

Eccolo invece in BASIC:

```
 5  REM un programma per moltiplicare
 6  REM due numeri
10  a = 3
20  b = 4
30  c = a * b
40  print "Il risultato è ";c
```

E in Pascal:

```
program add(input,output);

integer a,b,c;

(* un programma per moltiplicare due numeri *)

begin
  a := 3;
  b := 4;
  c := a * b;
  writeln('Il risultato:',c);
end.
```

Per prima cossa potete notare che un programma in C ha molta puntaggiatura. Il C utilizza la puntaggiatura nello stesso modo in cui Pascal utilizza ' begin ' e ' end '. Questo sembra rendere i programmi in C piu' difficili da leggere dei programmi in Pascal, per esempio. Mentre il Pascal utilizza ' begin ' e ' end ' per delimitare l'inizio e la fine delle procedure e delle funzioni, il C utilizza le parentesi graffe ' { ' e ' } '. Inoltre, il Pascal e il C separano i commenti all programma dal codice ancora con l'utilizzo della punteggiatura. Il C utilizza la coppia di ' /* ' e ' */ ', mentre il Pascal utilizza ' (* ' e ' *) '. Il BASIC utilizza invece lo statement REM all'inizio della linea per segnalare all'interprete un commento. Coloro che utilizzano AmigaBASIC sanno pero' che i commenti possono venire segnalati al compilatore anche attraverso l'utilizzo dell'apostrofo (') e che non sono necessari i numeri di linea. Quasi tutto cio' che viene detto a proposito del Pascal può essere applicato al Modula-2.

Sia il Pascal che il C sono linguaggi compilati. Osservate come il Pascal e il C dichiarino esplicitamente negli esempi le variabili a, b e c. Il Pascal scrive ' integer ' dove il C scrive ' int ' per dichiarare una variabile integera.

Questo e' comune ai linguaggi compilati. Un compilatore infatti ha bisogno di conoscere il tipo (type) della variabile prima di utilizzarla nel programma. Questa conoscenza verrà utilizzata per eseguire correttamente le operazioni, come addizioni o moltiplicazioni, quando questa variabile viene utilizzata successivamente, poiché numeri interi e numeri a virgola mobile vengono utilizzati differentemente e occupano un diverso spazio di memoria.

L'esempio in BASIC no necessita di dichirazioni di tipo esplicite per ciascuna variabile. Il BASIC è un linguaggio interpretato. (Naturalmente esistono anche ottimi compilatori per il BASIC come quello offerto dalla Microsoft per AmigaBASIC.) Quando voi digitate RUN, il computer analizza il testo di ciascuna linea per eseguire successivamente i diversi passi del programma. Se il BASIC incontra una variabile che non riconosce, asssume che quela variabile sia un numero in virgola mobile. Quindi nel nostro esempio a,b e c sono variabili in virgola mobile.

IL C ha pure anche altri tipi di variabili oltre a ' int ': ' float ' per virgola mobile, e ' char ' per variabili carattere e altre che vedremo a suo tempo. Ciascun tipo possiede un intervallo valido di valori. Per esempio. ' int ' e' limitato a più o meno due bilioni in Amiga, mentre ' char ' e' limitato ai valori tra +127 e -128. In realtà per questi intervalli bisogna sempre tenere anche conto delle implementazioni reali dei vari compilatori.

I tipi ' integer ', cioe' interi come ' char ' e ' int ', possono venire modificati dalla parola ' unsigned ' - senza segno -, creando in tal modo i tipi ' unsigned char ' e ' unsigned int '. Questi tipi hanno rispettivamente intervalli tra 0 e 255 e tra 0 e oltre i 4,2 bilioni. Se voi conoscete bit e byte, potrete dedurne che questi valori possono essere rappresentati in 8 e 32 bit. Ciascun tipo ha diversi operatori validi, come addizione e sottrazione, moltiplicazione e divisione. Diversamente dal BASIC e dal Pascal, il C e' molto flessibile relativamente alla possibilità di operare con essi. Per esempio, potete addizionare variabili ' char ' e variabili ' int '. Il valore di un ' char ' e' il valore ASCII del carattere memorizzato all'interno di esso. In BASIC potreste ottenere questo valore utilizzando la funzione ASC(), in Pascal potreste invece utilizzare ORD().

Avrete senz'altro osservato che le linee del programma in C e in Pascal terminano con un punto e virgola. Questa è un'altra caratteristica dei linguaggi compilati. Il ' ; ' comunica al compilatore che il programmatore ritiene che la linea sia completa. Infatti, le linee dei programmi in C possono essere distribuite su diverse linee fisiche senza alcuna conseguenza dannosa.

```
printf
(
"Il risultato e' %d"
,
c
)
;
```

Il compilatore analizza la linea nello stesso modo. E questo non è possibile con la maggior parte degli interpreti BASIC.

Incominciamo a prendere in considerazione il codice C.

```
#include "stdio.h"
```

È un comando per una parte speciale del compilatore chiamata preprocessore. Questa linea compilatore: "a questo punto compila anche il file programma chiamato stdio.h". Questo file contiene le dichiarazioni delle variabili e delle funzioni utilizzate da printf() e altre funzioni standard di I/O. Il preprocessore ha anche altre caratteristiche che considereremo in seguito.

Osserviamo ora attentamente la linea che contiene la funzione printf(). Printf() ha due argomenti, una stringa di caratteri contenente il testo "Il risultato è %d\n" è il nome di una variabile intera, c. Per un programmatore in C questi sono gli argomenti passati alla funzione, gli argomenti della funzione oppure ancora i valori passati alla funzione. Come avrete ormai capito gli argomenti vengono collocati tra le parentesi e separati dalle virgole.

Il primo argomento passato alla funzione printf() descrive il formato del testo che noi vogliamo visualizzare sullo schermo. Esso contiene due parti che potrebbero esserci poco familiari, entrambe presenti verso la fine del testo tra virgolette. (Ah, se voi conosceste il FORTRAN, potreste ora contare sulla conoscenza del comando FORMAT!) Il %d significa "qui stampa un valore intero". Il \n significa "qui stampa una carattere di accapo", che dovrebbe muovere il cursore sulla linea successiva, esattamente come succede quando voi premente il tasto del Return. Se noi desideriamo visualizzare due valori utilizzando printf(), possiamo scrivere:

```
printf( "Noi abbiamo %d mele e %d
        pere.\n",mele, pere);
```

Per ottenere questo risultato:

```
Noi abbiamo 7 mele e 5 pere.
```

Se naturalmente le variabili ' mele ' e ' pere ' contengono rispettivamente i valori 7 e 5.

Noi potremmo chiedere a printf() di stampare un valore in un campo di determinata ampiezza. Se si utilizza %4d come formato per gli interi, il numero sara' visualizzato con gli spazi necessari per riempire quattro posizioni carattere. La funzione printf() ha altri comandi di formattazione - per una visione complessiva e dettagliata consultare sempre il manuale del proprio compilatore -, come %x, per esempio, che stampa valori in esadecimale oppure %c, utilizzato con le stringhe formattate per visualizzare un valore come carattere ASCII, similmente al comando BASIC CHR$().

Le funzioni vengono utilizzate estensivamente in C; e alle funzioni pre-scritte che il linguaggio vi mette a disposizione ben presto si affiancheranno le funzioni che voi stessi scriverete per i vostri programmi. Se voi avete utilizzato il Pascal avrete acquistato una certa familiarità con le procedure e le funzioni. Ebbene le funzioni C ricordano proprio queste ultime dal momento che ritornano tutte un valore. (In realtà esiste la parola riservata VOID che preclude alle funzioni la possibilità di ritornare valori, ma di ciò in seguito, quando ne saprete molto di più.) Le funzioni C, comunque, non possono venire nidificate come succede nel Pascal con le procedure e le funzioni.

Se voi non avete mai utilizzato il Pascal, pensate alle funzioni BASIC SIN() e COS(). Queste funzioni ritornano un valore in virgola mobile basato sul loro argomento, e cioè il numero tra parentesi. Tutte le funzioni C ritornano un valore di un determinato tipo. Eccovi un frammento di programma che dichiara una funzione che ritorna un intero, il prodotto di due argomenti interi:

```
/* una funzione per moltiplicare due numeri */

int mult(a,b)
int a,b;
{
  int c; /* una variabile locale */

  c = a * b;
  return(c);
}
```

La prima linea dichiara una nuova funzione chiamata mult(), che ha due argomenti, a e b, entrambi interi. Una dichiarazione di funzione può essere suddivisa in diverse parti. Prima, il tipo della funzione. Qui, noi abbiamo utilizzato ' int '. Se nessun tipo viene dichiarato, il C assume che la nuova funzione sia di tipo ' int '. Seconda, il nome della funzione, ' mult '. Tutte le dichiarazioni delle funzioni hanno un'insieme di parentesi, sia che la funzione abbia argomenti oppure no. Questa funzione ha due argomenti. Il tipo di ciascun argomento presente tra le parentesi deve venir dichiarato prima della parentesi graffa aperta. La parentesi graffa aperta indica al compilatore l'inizio del codice della funzione.

La linea dopo la parentesi graffa aperta dichiara una variabile locale chiamata c. Se voi conoscete il Pascal, le variabili locali in C lavorano nello stesso modo che in Pascal, tenendo a mente pero' la restrizione che le funzioni non possono essere nidificate in C. Se voi, invece, non conoscete il Pascal, una variabile locale è una variabile a cui non si può accedere da un'altra procedura o funzione in un programma, anche se un'altra funzione dichiara una variabile locale con lo stesso nome. Il valore di una variabile locale è conosciuto solamente durante l'esecuzione di una variabile. Al termine dell'esecuzione della funzione, la variabile locale scompare, e la memoria che essa ha utilizzato vien riutilizzata.

Il ' c = a * b ' dovrebbe essere chiaro, qualsiasi linguaggio conosciate. La linea ' return(c); ' rappresenta la fine della funzione mult(). Il valore ' c ' viene ritornato alla funzione che ha chiesto di utilizzare la funzione mult(). Vediamo ora un programma completo in C che utilizza la funzione mult() descritta sopra.

```
#include "stdio.h"

/* una funzione per moltiplicare due numeri */

int mult(a,b)
int a,b;
{
  int c; /* una variabile locale */

  c = a * b;
  return(c);
}

/* Tutte le variabili dichiarate all'esterno delle
   dichiarazioni della funzione sono variabili
   globali, che possono essere utilizzate da
   tutte le funzioni
*/

int t; /* una variabile globale */

main()
{
int r,s; /* variabili locali */

r = 3;
s = 4;
t = mult(r,s);
printf(Il risultato e' %d\n",t);
}
```

In questo programma, la funzione main() chiama mult() con i valori memorizzati in ' r ' e ' s '. Ipotizzando che ' r ' contenga 3 e ' s ' contenga 4, la funzione mult() ritornera' il valore 12. All'interno della funzione main() il valore 12 verrà memorizzato nella variabile globale di main() chiamata ' t '.

Un programmatore spiegherebbe cio' che è accaduto dicendo che la funzione mult() è stata ' chiamata ' dalla funzione main() e il valore 12 e' stato ' ritornato ' a main(). Poinché il C e' basato su funzioni e chiamate di funzioni, i programmatori spesso utilizzano queste espressioni. Le espressioni ' chiamata ' e ' valore di ritorno ' alludono alla struttura fondamentale del C. Entrambe le espressioni vengono utilizzate nello stesso contesto per programmi in linguaggio macchina. E ora a main().

La dichiarazione della funzione main() non ha argomenti, e nessun tipo specifico vene dichiarato, in questo modo main() ritornera' un ' int '. Main() e' una funzione speciale. Quando il programma C viene avviato dal sistema operativo, main() è la prima funzione che viene eseguita. A parte questo, main() non è in alcun altro modo differente dalle altre funzioni C.

Ma ritorniamo alle veriabili. L'opposto di locale è globale. Le variabili globali sono tutte quelle variabili che vengono dichiarate all'esterno di tutte le funzioni del programma. D'altra parte le funzioni globali vengono dichiarate nello stesso modo delle variabili locali. La sintassi è la stessa: prima il tipo - int, per esempio -, quindi i nomi delle variabili, separati da virgole, se ce ne sono più di una, e per finire la terminazione ' ; '. Le variabili globali possono venire lette e modificate da qualsiasi funzione del programma. La maggior parte dei programmi presentano una mescolanza di variabili locali e variabili locali. Le variabili vengono utilizzate per calcoli intermedi, dal momento che scompaiono quando la funzione termina. Le variabili globali vengono utilizzate per i dati piu' permanenti, e per quei dati che diverse funzioni possono avar necessità di utilizzare. Se una funzione dichiara una variabile locale con lo stesso nome di una variabile globale, la variabile locale ha la precedenza.

Nel nostro prossimo incontro approfondiremo l'utilizzo del preprocessore, e come dichiarare altri tipi di variabili. Ci introdurremo nell'argomento fondamentale dell'I/O considerandone le funzioni a disposizione. E forse ci diletteremo con le funzioni speciali di Amiga per le creazione di window.

CORSO DI ASSEMBLY

Come si programma in Assembly

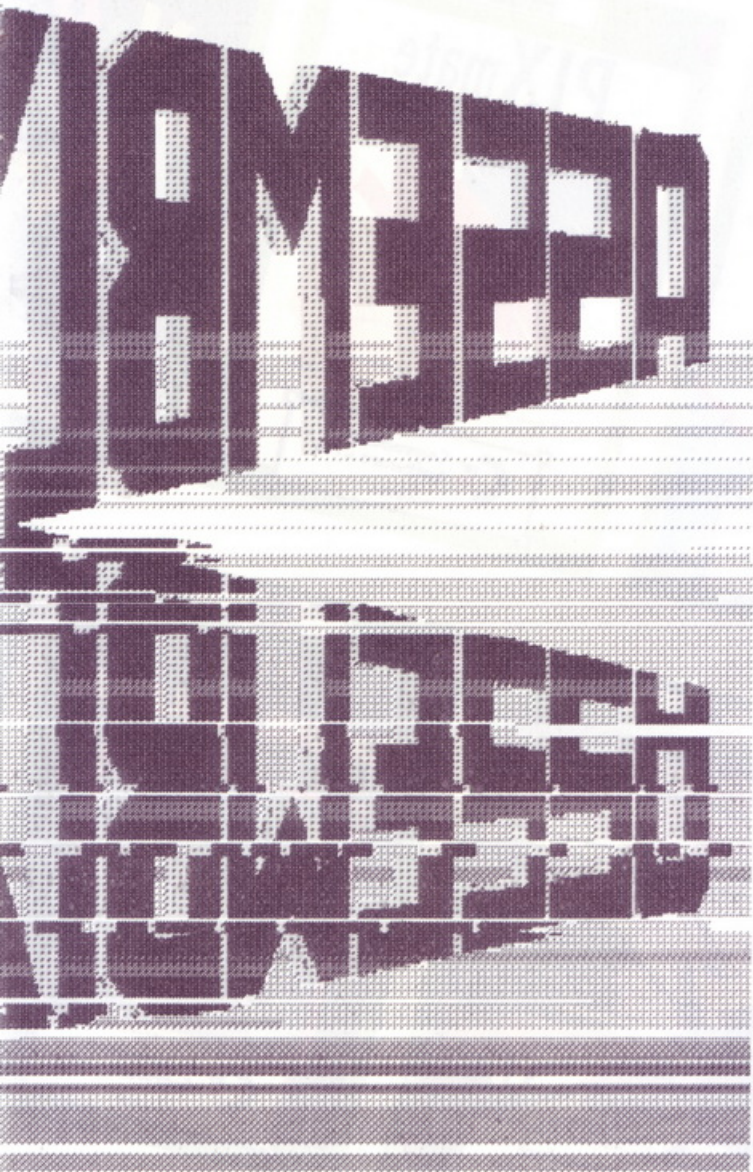
Il primo passo per realizzare un programma è, naturalmente, sedersi e pensare con calma a cosa si deve fare; è consigliabile poi scarabocchiare una prima versione del programma su carta. Si può quindi passare alla fase di editing: con l'ausilio di un qualunque editor, come il comando ED del CLI, si digita il programma e si salva il testo (anche un word processor va bene, purché capace di salvare in ASCII). Se si decide di usare ED bisogna entrare in CLI, richiamare l'editor digitando ED nomesorgente (dove nomesorgente è il nome che si decide di assegnare al testo, noto in gergo programmatorio come codice sorgente) e, dopo aver inserito tutto il testo, salvarlo con ESC X. Bisogna quindi convertire il testo così ottenuto in un programma in linguaggio macchina; questa operazione, che nell'ambito dei linguaggi evoluti è svolta da un interprete o da un compilatore, richiede invece nel nostro caso un assemblatore (più diffusamente noto come assembler). Ne esistono di molti tipi e in questa serie di articoli verrà impiegato il macroassembler standard prodotto dalla Metacomco; gli altri assembler possono differirne nell'uso delle macro, delle label locali (nn$) e delle direttive di assembly come CNOP 0,2 che serve ad allineare il programma a indirizzo pari: se quindi avete un assembler diverso leggete il relativo manuale. Se invece avete lo stesso assembler potete richiamarlo, sempre da CLI, con ASSEM nomesorgente -O nomeoggetto, dove quest'ultimo parametro è il nome con cui desiderate battezzare il programma assemblato, generalmente noto come codice oggetto. Occorre infine linkare il tutto con il comando ALINK nomeoggetto TO nomedefinitivo; ALINK si trova, assieme ad ASSEM, nella directory c del disco dell'assembler e se il CLI ha qualche difficoltà nel trovarlo dovrete specificare il pathname completo (ASSEM-DEVEL:c/ASSEM e ASSEM-DEVEL:c/ALINK) quando li attivate. Dopo tutta questa procedura avrete ottenuto un file chiamato nomedefinitivo che può essere mandato in esecuzione da CLI semplicemente digitandone il nome. Altri tipi di assembler possono avere l'editor e il linker incorporati (caratteristica che fece la fortuna del TurboPascal), allo scopo di semplificare la vita del programmatore. I possessori del sopracitato assembler della Metacomco sono vivamente consigliati di usare estesamente il RAM Disk e le sequenze di comandi in base alla propria esperienza individuale, per non estenuare in misura eccessiva il drive e la propria pazienza.

Se il vostro programma non funziona esiste un'ampia gamma di tool per il debugging come monitor e disassembler (in genere presenti nello stesso package); il primo vi consente di controllare da vicino lo stato dei registri del processore e delle locazioni di memoria durante l'elaborazione, permettendo di inserire dei breakpoint (istruzioni di stop temporaneo) nel programma o di eseguire un'istruzione alla volta (tracing); il secondo riconverte qualunque codice oggetto in qualcosa di abbastanza simile al codice sorgente originario, anche se si perdono tutte le label.

Struttura di una linea

L'Assembly non prevede l'uso del multistatement, cioè della possibilità di inserire più comandi in una stessa riga. Ogni linea di programma possiede una struttura ben precisa e può essere pensata come sud-

Seconda puntata del corso di programmazione sul linguaggio Assembly MC68000, il linguaggio macchina dell'Amiga

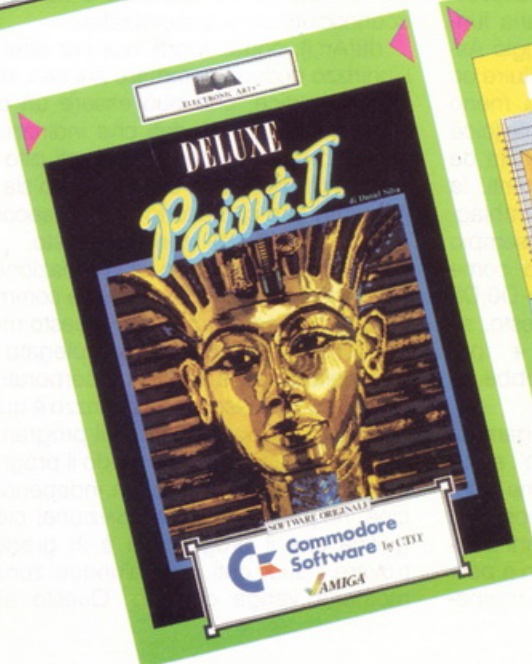
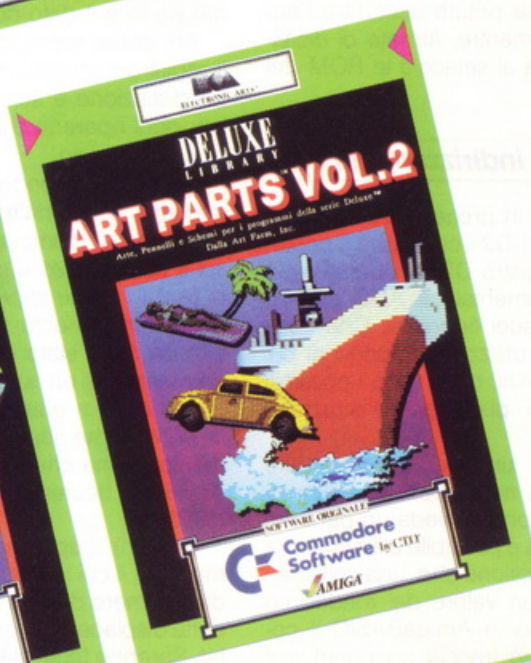
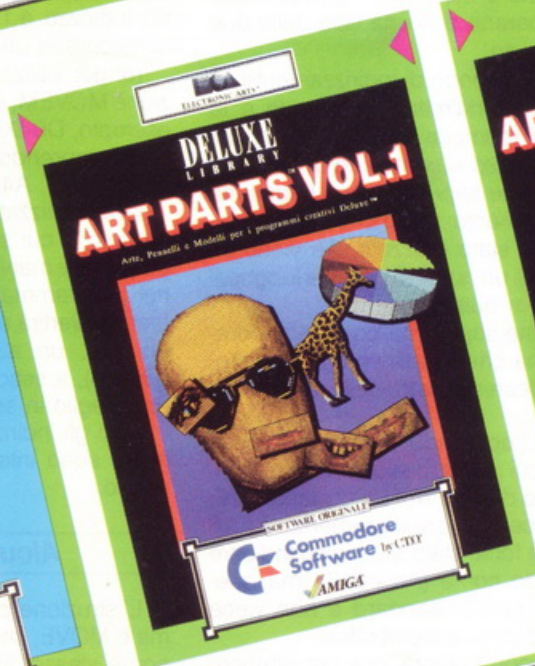
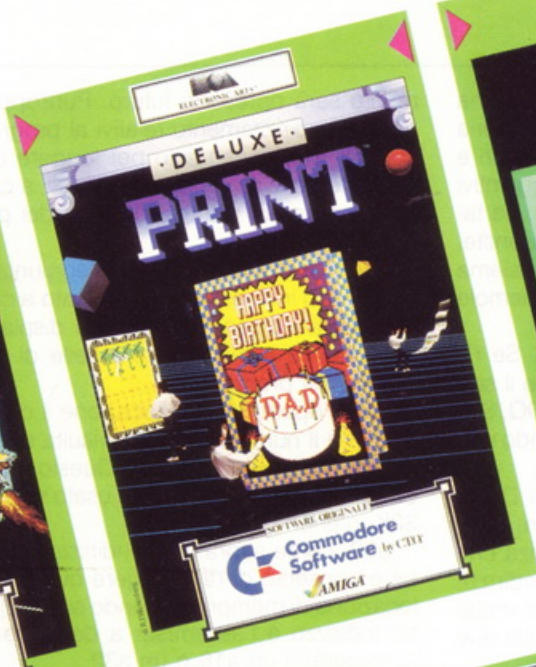
di Paolo Russo

divisa in campi, la lunghezza di ognuno dei quali può però variare da linea a linea. I campi sono quattro: label, istruzione, operandi e commento. Una label (etichetta) è una corta stringa alfanumerica usata per marcare un punto del programma, come riferimento per i salti e per la manipolazione delle variabili; è importante notare che le label esistono solo per l'assemblatore, non per il microprocessore. Se a esempio si contrassegna un punto del programma con una label e poi in un altro punto si inserisce un'istruzione di salto a quella etichetta l'assembler sostituirà alla label, nell'istruzione di salto, un numero che rappresenta l'indirizzo di memoria a cui saltare. Il microprocessore ragiona sempre in termini di indirizzi numerici: le label sono comode per gli esseri umani e per questo motivo l'assembler le adotta, convertendole poi in numeri e indirizzi al momento di generare il codice oggetto. Il campo della label può essere vuoto.

Il campo successivo, quello dell'istruzione, non può mai essere nullo e deve contenere il nome dell'istruzione stessa, mentre gli operandi, se presenti, devono essere inseriti nel campo successivo, separati l'uno dall'altro da una virgola (non possono esservene più di due). Il commento finale è opzionale. Si può, se lo si desidera, riservare un'intera linea a scopo di commento, inserendo nella prima colonna un carattere particolare che in genere è un asterisco ma che può in effetti variare da assembler ad assembler. I vari campi devono essere separati da almeno uno spazio; questo significa che anche se in una linea non c'è alcuna label occorre comunque lasciare almeno uno spazio prima di digitare il nome dell'istruzione, in modo da non confondere l'assembler.

Byte, word e long word

I registri del 68000, come illustrato nella scorsa puntata, sono a 32 bit, ossia sono delle long word. Nonostante ciò, non sempre il processore esegue calcoli con 32 bit: per motivi di ordine pratico ci si limita spesso a 16 oppure 8 bit. La lunghezza dei dati su cui si opera non è intrinseca al dato stesso ma deve essere specificata in ogni istruzione. Per esempio esistono tre forme dell'istruzione ADD: ADD.B, ADD.W e ADD.L che sommano rispettivamente numeri a 8, 16 e 32 bit. Naturalmente B, W ed L significano rispettivamente byte, word e long word. L'istruzione ADD.B D1,D2 somma gli otto bit meno significativi di D1 e D2, ponendo il risultato negli otto bit meno significativi di D2. I rimanenti 24 bit di D2 non vengono alterati; eventuali riporti da e verso il nono bit vengono troncati senza pietà.

Quando non è presente alcun suffisso l'assembler parte solitamente dal presupposto che esista un .W sottinteso; in altre parole ADD D1,D2 significa ADD.W D1,D2.

Quando si lavora con i registri occorre tenere presente che ogni operazione su byte e word agisce sempre sulla parte meno significativa del registro coinvolto. I registri indirizzi si comportano in maniera alquanto originale: per motivi tecnici è vietato operare su di loro con una lunghezza di soli otto bit; se si tenta di scrivervi, sommarvi o sottrarvi un numero a soli 16 bit questo viene automaticamente esteso a 32 bit e l'operazione coinvolge poi l'intero registro. Esempio: ADD.B #10,A0 è illegale, ADD.W #10,A0 e ADD.L #10,A0 agiscono entrambe su 32 bit. L'unica eccezione a questa regola è costituita dalle istruzioni ADDQ.W #n,An e SUBQ.W #n,An che agiscono sulla sola word inferiore di An.

Questa è un'importante eccezione che è opportuno ricordare, in quanto capita sovente di incrementare o decrementare un registro indirizzi con ADDQ e SUBQ, con l'intenzione di agire su long word, e di dimenticare il .L nella illusoria consapevolezza che tutte le operazioni sui registri indirizzi avvengano a 32 bit. Il bug che ne deriva colpisce sporadicamente (solo quando si verifica un riporto da o verso la word superiore, che resta invece inalterata) ed è quindi assai arduo da snidare se non se ne sospetta l'esistenza. Questo bug è così sfuggente che non è troppo raro trovarlo addirittura in qualche sistema operativo, come ha potuto accertare l'autore dell'articolo mentre, armato di disassembler, passava al setaccio la ROM del suo QL.

I modi di indirizzamento

Abbiamo visto in precedenza come l'istruzione ADD D1,D2 sommasse il contenuto di D1 al registro D2; in questo caso l'istruzione fondamentale è ADD mentre D1 e D2 sono i suoi operandi. Che cosa avremmo potuto utilizzare in luogo di D1 e D2? In altri termini, quali sono i possibili operandi su cui può agire l'istruzione ADD?

Nell'ambito di un linguaggio evoluto questa domanda non avrebbe senso: ovunque un'istruzione richieda un parametro di un certo tipo è possibile utilizzare una qualunque espressione che, una volta calcolata, fornisca un valore del medesimo tipo. Per esempio in AmigaBASIC il comando PSET (x,y) traccia un punto alle coordinate x e y, dove questi due parametri possono essere costituiti da espressioni a valore numerico di arbitraria lunghezza e complessità. Nel linguaggio Assembly invece non si può mai sostituire un parametro con un'espressione, a meno che il parametro non sia una costante (cioè un numero), che può essere sostituita da una espressione formata da costanti, la quale verrà calcolata in fase di assemblaggio e non in fase di run-time. A esempio ADD #13,D5 può essere scritto come ADD #8+5,D5 mentre ADD D1+D0,D2 non ha senso in Assembly. Del resto, se fosse sufficiente usare il segno '+' per eseguire un'addizione non ci sarebbe alcun bisogno dell'istruzione ADD.

In Assembly esiste una ristretta gamma di forme che un'espressione puo' assumere perché sia lecito usarla come operando, e quel che è peggio è che non tutte queste forme sono ammesse in ogni istruzione. In un linguaggio evoluto non è possibile stimare esattamente quanta memoria occupa una linea di programma, ma in Assembly sì: l'occupazione di memoria di una singola istruzione è sempre pari a due byte più il numero di byte aggiuntivi richiesti da ognuno degli operandi. La tabella 1 mostra le espressioni consentite, chiamate modi di indirizzamento, assieme al loro consumo di memoria; passiamole brevemente in rassegna:

#n: l'operando è il numero n. Se l'istruzione è veloce (quick), ossia se il suo nome termina per Q (MOVEQ, ADDQ, SUBQ) l'uso di #n come primo operando non richiede byte aggiuntivi.

Dn: l'operando è il registro dati Dn, o meglio lo è il dato contenuto in Dn.

An: come sopra, con la differenza che il registro coinvolto è un registro indirizzi. La distinzione è sensata in quanto molte istruzioni operano con una sola delle due classi di registri.

(An): questo modo di indirizzamento e i seguenti sono indiretti. L'operando è la locazione di memoria puntata da An, ossia il cui indirizzo è contenuto in An.

(An)+: come sopra, ma con l'aggiunta del postincremento: dopo che il dato puntato da An è stato utilizzato, An viene incrementato di un numero pari alla lunghezza del dato in questione.

-(An): come sopra, ma con il predecremento: prima che il dato venga utilizzato, An viene decrementato della lunghezza del dato.

d16(An): l'operando è la locazione di memoria il cui indirizzo si ottiene sommando il numero d16 ad An. La costante d16, detta displacement o spiazzamento, è a 16 bit. Sarebbe forse stato più logico indicare questo modo come (An+d16), ma questo formato non è standard e non viene riconosciuto dagli assemblatori.

d8(An,i): come sopra, ma per ottenere l'indirizzo occorre sommare anche i, detto registro indice, che può essere un qualunque registro, sia dati che indirizzi. Lo spiazzamento ammesso è a soli otto bit, mentre l'indice i può essere inteso sia come word che come long word a seconda del suffisso che gli viene posposto.

d16(PC): l'operando è la locazione di memoria il cui indirizzo si ottiene sommando d16 al program counter. Questo modo di indirizzamento è spesso impiegato per manipolare dati e variabili incorporati nel programma stesso e il cui indirizzo è quindi vincolato alla posizione che il programma occupa nella RAM. In tal modo il programma che si ottiene è position independent, cioè indipendente dalla posizione; ciò significa che il programma è in grado di trovare i suoi dati in qualunque zona di memoria venga caricato. Questo argomento sarà ripreso in futuro. Purtroppo i modi di indirizzamento relativi al program counter funzionano solo per leggere dati dalla memoria e non per scriverveli, a causa di una assai discutibile scelta dei progettisti.

d8(PC,i): come sopra, con l'aggiunta di un registro indice. Non molto usato a causa della limitazione imposta sul displacement, serve a manipolare tabelle di dati incluse nel programma.

a16: l'operando è la locazione il cui indirizzo è il numero a16, che risulta quindi essere un indirizzo a 16 bit. Questo modo di indirizzamento può essere usato solo nei 32K inferiori di RAM.

a32: come sopra, ma l'indirizzo è a 32 bit e consente di raggiungere qualunque locazione di memoria. Quando si specifica un indirizzo è l'assembler a decidere se considerarlo un a16 o un a32.

range: utilizzato unicamente dall'istruzione MOVEM, è un insieme di registri. Per esempio, D0-D3/D5/A1/A3-A5 è un range che comprende i registri D0, D1, D2, D3, D5, A1, A3, A4 e A5.

label: utilizzata nei salti, per specificare il punto a cui si salta. Tramite le label si può anche manipolare un dato incorporato nel programma: in questo caso l'assembler convertirà automaticamente la label in un a16, un a32, un d16(PC) o in un d8(PC,i), a seconda dei casi, come si vedrà meglio in seguito.

Tutti gli indirizzi, gli indici e gli spiazzamenti sono intesi come numeri dotati di segno.

Alcune istruzioni

L'istruzione più usata in ogni programma è MOVE, che, come si può arguire dal nome stesso, sposta dati da un punto a un altro e può essere considerata un'istruzione di assegnazione, simile all'arcaico LET di alcuni dialetti BASIC. Il comando MOVE richiede due operandi che devono essere separati, com'è d'uso in Assembly, da una virgola; il primo di essi, detto sorgente, fornisce il valore da assegnare al secondo, detto destinazione. Per esempio MOVE.W D1,D2 copia la word meno significativa di D1 in quella di D2, senza alterarne la metà superiore; MOVE.L D0,A3 assegna il contenuto di D0 ad A3; MOVE.B #10,D6 assegna il valore dieci al byte inferiore di D6 mentre MOVE.B 10,D6 legge il byte di memoria all'indirizzo dieci e lo pone in D6 (attenti al '#' !). MOVE e' l'istruzione che consente la massima libertà nella scelta dei modi di indirizzamento per gli operandi; quasi ogni combinazione di modi è lecita. La maggior parte delle altre

istruzioni consente tale libertà per uno solo dei due operandi, quello sorgente o quello destinazione, a scelta del programmatore, mentre l'altro parametro può variare in un ambito molto limitato (solitamente #n e Dn, ma non sempre). A esempio ADD.W D1,D1, ADD.W 16(A3),D1 e ADD.W D1,16(A3) sono lecite mentre ADD.W 16(A3),16(A3) non lo è; MOVE.W 16(A3),16(A3) è invece perfettamente lecita, per quanto improduttiva in questo caso particolare data la coincidenza dei due operandi. Tutto questo può sembrare un po' complicato ed eccessivamente mnemonico, ma l'esperienza insegna che il programmatore medio apprende in breve tempo con l'esercizio le combinazioni istruzione-operandi proibite e le rare volte che commette un errore l'assembler stesso lo segnala prontamente.

Esiste poi una piccola gamma di istruzioni di uso comunissimo che non consentono quasi nessuna libertà nella scelta dei parametri: a esempio MOVEQ #n,Dn, che richiede assolutamente un dato immediato (a otto bit, che viene esteso a 32, cosa questa atipica per un MOVE) come sorgente e un registro dati come destinazione. Una curiosità: la Q sta per quick, ossia veloce, in quanto la suddetta istruzione consuma poca memoria ed è caratterizzata da un basso tempo di esecuzione; in altre parole essa è di uso molto pratico. le istruzioni che limitano fortemente la scelta degli operandi sono tra le più usate; accade quindi qualcosa di simile al modo in cui alcuni linguaggi umani, come l'italiano e l'inglese, gestiscono i verbi: tra di essi, quelli irregolari sono quasi sempre i più usati.

Un operando capace di assumere qualunque forma viene chiamato indirizzo effettivo (effective address) e abbreviato in ea; per esempio la forma generale dell'istruzione di trasferimento è MOVE ea,ea, per sottolineare il fatto che non esiste quasi nessuna limitazione nella scelta dei modi di indirizzamento; l'istruzione di addizione può invece assumere una delle seguenti forme: ADD ea,Dn - ADD ea,An - ADD #n,ea - ADD Dn,ea. Anche l'addizione, come l'assegnazione, possiede una forma 'irregolare' veloce, che può assumere solo la forma ADDQ #n,ea dove n deve essere compreso tra uno e otto (viene però esteso a 8, 16 o 32 bit in base al suffisso .B, .W o .L). Questa simpatica istruzione è l'equivalente 68000 della meno flessibile INC di altri microprocessori.

Un piccolo esempio

Fino a questo momento non sono ancora state presentate sufficienti nozioni da consentire la realizzazione di un vero e proprio programma. Si può però prendere in considerazione qualche microscopico esempio. Il problema è il seguente: esistono tre interi a 16 bit in memoria agli indirizzi consecutivi 100000, 100002 e 100004, che per brevità indicheremo come A, B e C; bisogna porre nel terzo la somma dei primi due. Esistono svariati modi per farlo e ne prenderemo in considerazione alcuni.

PROGRAMMA 1	PROGRAMMA 2	PROGRAMMA 3
MOVE A,D0	MOVE A,D0	MOVE.L #A,A0
MOVE B,D1	ADD B,D0	MOVE (A0)+,D0
ADD D1,D0	MOVE D0,C	ADD (A0)+,D0
MOVE D0,C		MOVE D0,(A0)

Il primo metodo è il meno efficiente (forse il compilatore di un linguaggio evoluto lo sceglierebbe), il secondo è il più logico e intuitivo mentre il terzo, sfruttando il fatto che i dati in memoria sono consecutivi, è il più compatto e veloce. Non assemblate realmente questi programmi, se lo faceste otterreste una Guru Meditation dovuta all'assenza di un'istruzione di ritorno e alla arbitraria e distruttiva alterazione della word 100004. Se anche per assurdo tutto funzionasse non vedreste comunque nulla sullo schermo. Prossimamente, quando sarà stato illustrato l'uso dei flag e delle istruzioni di controllo, sarà possibile realizzare qualche programma vero, anche se breve.

SIMBOLO	N. BYTE Q B W L	NOME
#n	0 2 2 4	immediato
DN	/ 0 0 0	diretto a registro dati
An	/ 0 0 0	diretto a registro indirizzi
(An)	/ 0 0 0	indiretto a registro
(An)+	/ 0 0 0	indiretto a registro con postincremento
—(An)	/ 0 0 0	indiretto a registro con predecremento
di6(An)	/ 2 2 2	indiretto a registro con offset
d8(An,i)	/ 2 2 2	indiretto a registro indicizzato
d16(PC)	/ 2 2 2	relativo al program counter
d8(PC,i)	/ 2 2 2	relativo al program counter, indicizzato
a16	/ 2 2 2	assoluto corto
a32	/ 4 4 4	assoluto lungo
range	/ / 2 2	lista di registri
label	(0,2,4)	label

Se una label segue l'istruzione Bcc, DBcc e BSR allora viene convertita in un offset a 16 bit (n.byte=2) se invece segue l'istruzione Bcc.S o BSR.S allora viene convertita in un offset a 8 bit (n.byte=0) altrimenti:

label	viene convertito in a16 o a32	(n.byte=2,4)
label (PC)	viene convertito in d16(PC)	(n.byte=2)
label (PC,i)	viene convertito in d8(PC,i)	(n.byte=2)

Tabella 1: modi di indirizzamento del 68000.

CORSO DI

Seconda puntata del corso di programmazione sull'AmigaBasic, l'interprete Basic dell'Amiga

di Paolo Russo

La strutturazione è una componente molto importante di ogni linguaggio e la trattazione di tale argomento, già iniziata in precedenza, viene adesso conclusa con la presentazione di ulteriori informazioni di carattere eminentemente pratico, più che teorico, riguardanti sia le strutture di controllo che i subprogram. Viene poi discusso il problema della gestione della memoria dell'Amiga dal punto di vista del programmatore AmigaBasic; infatti, al contrario della maggior parte delle macchine a otto bit e di molte di quelle a sedici bit, che non soffrono del problema della condivisione delle risorse tra più programmi, l'Amiga possiede un sistema operativo multitasking che consente sì a più di un programma di girare contemporaneamente sulla macchina ma obbliga i programmi stessi ad attenersi a una serie di regole per evitare che due di loro tentino di usufruire della stessa zona di memoria. Protocolli simili esistono anche per quanto riguarda la condivisione della tastiera, del drive e di parecchie altre risorse, ma l'interprete AmigaBasic o il sistema operativo solitamente si occupano autonomamente di questi dettagli senza scomodare il programmatore. La gestione della RAM è invece lasciata agli esseri umani, essendo spesso la quantità di memoria disponibile un fattore critico per il funzionamento di un programma, dal momento che, al contrario di altre risorse come la tastiera e il drive, presenta spesso la seccante tendenza a esaurirsi.

Le strutture di controllo

L'AmigaBasic ha in comune con il Basic standard il classico e universalmente conosciuto ciclo FOR-NEXT, sul quale non e' opportuno soffermarsi a causa della sua eccessiva fama. Il ciclo WHILE-WEND, molto di moda recentemente e spesso impiegato anche a sproposito, è una generalizzazione del ciclo FOR-NEXT dal quale differisce nella condizione di uscita dal loop; mentre il ciclo FOR-NEXT si interrompe solo quando il valore di una certa

variabile supera un certo limite, il ciclo WHILE-WEND prosegue fintantoché la condizione (formalmente simile a quella che segue un IF) specificata dopo la parola WHILE si mantiene vera. Ciò significa che la linea WHILE 1=1: PRINT"X": WEND scriverà un numero infinito di X mentre la linea WHILE 1=2: PRINT"X": WEND non ne stamperà nemmeno una. La coppia WHILE-WEND consente anche di emulare quella FOR-NEXT; le seguenti due linee sono infatti equivalenti:

```
FOR I=1 TO 10: NEXT I=1: WHILE I<=10:
I=I+1: WEND
```

S'intende che il contrario non è possibile: un generico WHILE-WEND non è sostituibile con un FOR-NEXT. Qualcuno si chiederà che scopo abbia tentare di usare una struttura al posto di un'altra. Naturalmente, come risulta evidente dall'esempio riportato poco sopra, il rimpiazzamento di un ciclo FOR-NEXT con uno WHILE-WEND non è mai conveniente in termini sia di praticità che di leggibilità, ma un rispettabile numero di persone in vena di semplificazioni ritengono che due strutture al posto di una siano ridondanti ed evitano sistematicamente il FOR-NEXT nei loro programmi, tentando nel contempo con tutte le loro energie di convincere chiunque altro che ciò sia indice di buona programmazione. Il lettore è consigliato di non lasciarsi trasportare ciecamente dai flussi e riflussi della moda, esistenti nella programmazione come in molte altre attività umane, ed è altresì vivamente incoraggiato alla creazione di uno stile personale, soprattutto se non desidera che l'eccitante attività programmatoria si trasformi rapidamente in una grigia routine.

Al classico IF-THEN(-ELSE) è stato affiancato il più flessibile IF-THEN (-ELSEIF-THEN) ... (-ELSEIF-THEN) (-ELSE) -END IF che consente di eseguire in maniera condizionata interi blocchi di linee alla volta, che possono a loro volta contenere altri IF. L'interprete identifica la variante di IF (classica o moderna) che il programmatore intende usare basandosi sulla presenza o meno di istruzioni in linea subito dopo il THEN; se infatti tale parola chiave è seguita da una o più istruzioni l'interprete decide che si tratta di un IF classico, a linea singola; se al contrario dopo il THEN non si scrive niente deve evidentemente trattarsi della variante evoluta multilinea e l'AmigaBasic parte dal presupposto che le linee successive debbano essere eseguite solo se la condizione specificata risulta vera. Naturalmente è necessaria l'esistenza di una parola chiave per contrassegnare la fine del blocco di linee da eseguire condizionatamente; tale parola, che deve occupare un'intera linea, è END IF. La parola ELSE, che esige anch'essa di non essere preceduta o seguita da alcunché, marca invece l'inizio delle linee che devono essere eseguite solo se la condizione è falsa. Al contrario delle parole IF, THEN ed END IF, la cui presenza è obbligatoria, l'uso di un blocco introdotto da ELSE è opzionale, e altrettanto può dirsi per ELSEIF, utile in caso di scelte multiple. La parola ELSEIF, che richiede una condizione e un THEN dopo di sè, introduce un blocco di istruzioni che verranno eseguite solo se l'ultima condizione specificata è vera e tutte le precedenti sono false. Eccone un esempio:

```
IF A=1 THEN
  PRINT"UNO"
ELSEIF A=2 THEN
  PRINT"DUE"
ELSEIF A=3 THEN
  PRINT"TRE"
ELSE
  PRINT"NON È UNO, DUE O TRE"
END IF
```

L'indentazione, da qualcuno chiamata dentellatura, ossia l'abitudine di lasciare all'inizio di ogni riga uno spazio variabile in modo da evidenziare i blocchi, non è obbligatoria ma molti la consigliano per motivi stilistici, trovandola evidentemente molto estetica o particolarmente giovevole alla chiarezza del listato. Purtroppo quando si esce dall'ambito dei programmi banali accade che all'interno di ogni singola procedura il numero di IF, FOR e WHILE concatenati sia solitamente abbastanza elevato da far sì che alcune parti del listato dimostrino una spiccata tendenza a uscire dall'area visibile dello schermo a causa della spaziatura iniziale.

L'AmigaBasic possiede anche il GOTO, ma tale direttiva è stata implementata nella presente release dell'interprete in una forma non troppo flessibile; è infatti proibito saltare dentro e fuori da una struttura tramite GOTO, cosa invece consentita da qualche altro computer. Non è quindi possibile saltar fuori neppure da un banalissimo ciclo FOR-NEXT prima del suo esaurimento: se si desidera farlo occorre assegnare il valore limite alla variabile indice e saltare al NEXT, cosa questa fattibile ma indubbiamente alquanto scomoda.

I dati dei subprogram

È stato detto nella scorsa puntata che le variabili di un subprogram esistono solo al suo interno, eccezion fatta per quelle dichiarate come SHARED che sono condivise con il programma principale. La parola STATIC che compare nella prima linea di ogni subprogram ha il seguente significato: tutte le variabili locali al subprogram sono statiche, ossia il loro contenuto non viene perso tra una chiamata e l'altra del subprogram stesso.

In realta' la presenza della parola STATIC nella definizione di un subprogram è obbligatoria e ciò significa che l'AmigaBasic non supporta le variabili locali non statiche. Quest'ultimo tipo di variabile è utile per limitare l'occupazione di memoria e per implementare algoritmi ricorsivi, cioè per consentire che un subprogram possa richiamarsi da solo; ciò in AmigaBasic non è quindi possibile, a meno di ideare un meccanismo che salvi le variabili prima di ogni chiamata. Esiste tuttavia un ulteriore inconveniente legato alla staticità delle variabili: com'è possibile utilizzare degli array locali in un subprogram? Se dimensioniamo l'array in questione nel programma principale perdiamo il vantaggio della località, oltre a essere costretti a dichiararlo SHARED; se il DIM compare invece all'inizio del subprogram i guai iniziano quando il suddetto viene richiamato per la seconda volta e l'interprete si trova nella seccante situazione di dover dimensionare un array già esistente. Ciò è illegale in AmigaBasic e provoca il blocco del programma con segnalazione di errore. Come risolvere questo problema?

Sfogliando il manuale qualcuno avrà notato l'esistenza di un utile comando chiamato ERASE che cancella un array eliminandolo dalla memoria come se non fosse mai esistito. Si può quindi pensare di inserire questo comando alla fine di un subprogram in modo che alla successiva chiamata l'array, assente, possa essere ridimensionato. Per qualche strano motivo questo trucco non sembra funzionare molto bene; probabilmente il comando ERASE è stato creato appositamente per gli array globali e non funziona sempre su quelli locali. Comunque, anche se funzionasse regolarmente sarebbe un metodo troppo inefficiente, in quanto la cancellazione e successivo dimensionamento sono operazioni alquanto lente, ed eseguirle ogni volta che il subprogram viene richiamato non sembra essere la soluzione migliore. Un trucco più valido consiste nell'inserzione all'inizio del subprogram di una linea come la seguente:

```
IF FLAG=0 THEN FLAG=1:DIM AR(100)
```

La prima volta che il subprogram viene richiamato le sue variabili non esistono an

cora; FLAG vale quindi zero, come tutte le variabili non ancora inizializzate. Alla suddetta variabile viene quindi assegnato il valore uno e l'array viene dimensionato. A ogni successiva chiamata del subprogram la condizione espressa dopo l'IF risulterà falsa e non sarà quindi effettuato alcun tentativo di dimensionare nuovamente l'array. Se poi all'interno del programma principale viene eseguito un CLEAR tutte le variabili spariscono, array compreso, ma anche FLAG torna a zero e il trucco continua a funzionare.

Lo statement CLEAR

L'istruzione CLEAR in ogni versione di Basic provoca la cancellazione di tutte le variabili, ma in alcuni dialetti essa assume un significato piu' ampio. Nel caso in questione la suddetta istruzione consente di cambiare il modo in cui la RAM viene ripartita tra le sue tre aree fondamentali: lo heap, lo stack e il segmento del Basic. Lo heap (cumulo) è una sorta di magazzino pubblico di memoria inutilizzata al quale tutti i programmi attingono secondo le loro necessità, dopo aver rivolto una regolare richiesta al sistema operativo (a Exec, per la precisione); ormai nemmeno i programmi per computer riescono a salvarsi dalla burocrazia. Se non altro Exec non insabbia mai le pratiche e si preoccupa di sbrigarle nel giro di qualche millisecondo, respingendole, in un tempo altrettanto breve, solo se la memoria è ormai esaurita o non possiede le caratteristiche richieste: un traguardo di efficienza del tutto fantascientifico per la burocrazia italiana. Purtroppo certe richieste che devono essere rivolte alle sezioni competenti del sistema operativo per ottenere determinati risultati prevedono la compilazione, da parte del programma richiedente, di un vero e proprio modulo di complessità tale da far invidia al tristemente noto 740; ma di questo si tratterà meglio in futuro. Il numero che compare nella barra superiore dello Workbench indica appunto la quantità di memoria ancora disponibile nello heap. Ogni programma che gira sull'Amiga possiede uno stack (pila, catasta) il cui scopo è la memorizzazione temporanea di indirizzi e variabili locali e la cui dimensione varia solitamente tra uno e quattro kappa di RAM; l'interprete AmigaBasic è esso stesso un programma e possiede di conseguenza uno stack. Il segmento Basic è quella zona di memoria che l'interprete ha sottratto allo heap per destinarla alla memorizzazione del programma Basic e delle sue variabili. La sintassi completa del comando è CLEAR (,segmento (,stack)), dove le parentesi indicano, come già in precedenza, un parametro che può essere omesso.

L'esperienza insegna che praticamente nessuno trova utile alterare la dimensione dello stack, mentre molti programmatori desiderano aumentare l'esiguo spazio a disposizione del Basic, che normalmente ammonta a circa 25K. Bisogna innanzitutto premettere che non è buona norma assegnare troppa RAM al segmento lasciando al verde lo heap, non solo perché risulterebbe impossibile il funzionamento in multitasking di un qualunque altro programma assieme all'interprete, ma anche perché determinate azioni intraprese da un programma Basic assorbono memoria dallo heap; il comando SCREEN per esempio definisce un nuovo schermo e preleva dallo heap la memoria necessaria alla relativa pagina grafica. Quando inoltre vengono eseguite certe operazioni, come lo spostamento di una finestra con il mouse, il computer necessita temporaneamente di una certa quantità di RAM per portare a termine il compito richiesto e se non la trova possono accadere stranezze di vario genere.

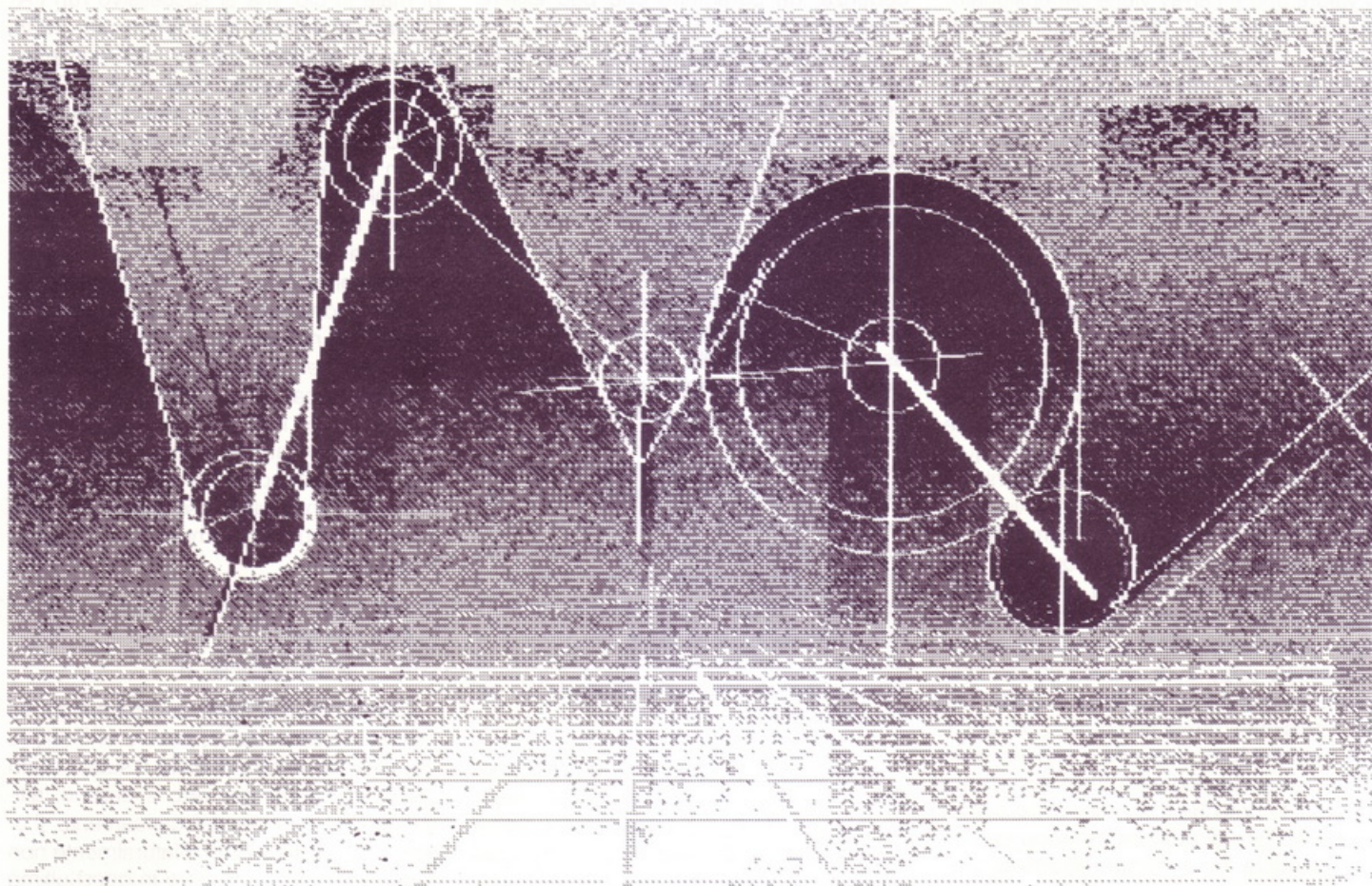
Il modo più semplice per riservare al segmento una certa quantità di RAM consiste nell'eseguire uno statement come CLEAR ,100000 (se si desiderano circa 100K; notate comunque la curiosa quanto obbligatoria presenza della virgola). Questo metodo presenta qualche inconveniente; il primo consiste nell'impossibilità di allocare ripetutamente più di metà (nel migliore dei casi) della memoria disponibile. Supponiamo per esempio che su di un Amiga inespanso si desideri portare il segmento a 120K; basta digitare in modo diretto CLEAR ,120000 e si è poi liberi di creare programmi con array di dimensioni veramente notevoli. Quando poi il programma viene salvato e in seguito ricaricato, magari dopo un reset, si scoprirà che non funziona più perché l'effetto del comando CLEAR è svanito, dal momento che anche l'interprete è stato ricaricato da disco, e non c'è più posto per le variabili. Evidentemente è opportuno inserire il CLEAR in testa al programma stesso, in modo che venga eseguito a ogni caricamento; così facendo, tuttavia, tale comando verrà eseguito anche a ogni run e ciò crea un altro problema, connesso al modo in cui l'interprete porta a termine un CLEAR. L'AmigaBasic richiederà i 120K prima di restituire i vecchi 25K allo heap, altrimenti il programma Basic verrebbe perso per strada. Per un attimo quindi l'interprete possiederà 145K. Non c'è niente di male in ciò, ma al successivo run l'AmigaBasic tenterà di ottenere altri 120K prima di mollare i vecchi 120, con esito infelice non essendo disponibili 240K di memoria. Si può allora ricorrere a un doppio CLEAR; supponendo che il programma in sè (variabili escluse) non richieda più di 10K, si può eseguire un CLEAR ,10000:CLEAR ,120000. In tal modo la massima occupazione di RAM in ogni dato istante non supera 130K. Esiste poi un ulteriore inconveniente: se l'abbondanza di memoria è necessaria non solo per le variabili, ma per il programma stesso, risulta impossibile caricarlo in memoria prima di aver eseguito il CLEAR, che a sua volta entrerebbe in azione solo dopo il caricamento: il classico serpente che si morde la coda. L'unica via di uscita consiste nella creazione di un programma di lancio come il seguente:

```
CLEAR ,5000
CLEAR ,120000
LOAD "nomeprogramma",R
```

Questo microscopico programmino, se lanciato in esecuzione, riserva al Basic la quantità desiderata di RAM e subito dopo carica il programma vero e proprio, attivandolo al termine del caricamento (l'opzione R significa Run). Naturalmente non è piu necessario includere in quest'ultimo programma istruzioni per allocare memoria, essendo questo compito già svolto dal programma di lancio.

La funzione FRE(x) consente di conoscere in ogni momento il modo in cui la RAM è ripartita; il classico FRE(0) fornisce la quantità di memoria ancora libera all'interno del segmento Basic, FRE(-1) la dimensione dello heap, FRE(-2) la quantità di RAM nello stack che non è mai stata usata. Ogni volta che la funzione FRE viene invocata l'interprete esegue il cosiddetto garbage collecting (alla lettera, raccolta di immondizia); tale operazione riordina le stringhe presenti in memoria in modo da riutilizzare gli interstizi che si formano di solito durante la loro manipolazione.

Una piccola curiosità: anche il comando CLEAR del vecchio ZX Spectrum ha la capacità di variare la porzione di RAM riservata al Basic! Chissà quanti altri computer possiedono questa caratteristica così assolutamente non standard associata al comando CLEAR?

POLINOMI A TRATTI?

SPLINE!

Vediamo come sfruttare al meglio gli algoritmi visti nel numero precedente, presentando così il concetto di polinomi a tratti.

Analizzeremo poi pregi e difetti delle migliorie introdotte fino ad individuare quella famiglia di curve detta spline, le cui proprietà interesseranno certamente non solo chi ha problemi di estrapolazione, ma anche tutti coloro che desiderano raccordare con una "dolce" curva i punti di un disegno.

Problema di instabilità

Il problema dell'interpolazione è, come abbiamo visto, quello di trovare una funzione che simuli abbastanza bene un'altra più o meno nota.

Vediamo di precisare cosa si intenda per "abbastanza bene": ciò che si richiede alla nostra funzione è che in corrispondenza ad ogni valore dell'incognita x assuma un valore f(x) poco differente da quello che assumerebbe la funzione che si vuole simulare.

Non sembri assurdo che si parli di valore della funzione da interpolare anche quando quest'ultima è nota solo in alcuni punti: infatti, anche se le informazioni quantitative si limitano a qualche nodo, generalmente ne conosciamo qualitativamente il comportamento fra un nodo e l'altro. Proprio l'idea sull'andamento della funzione interpolanda ci mette in guardia dalle eventuali anomalie della funzione interpolatrice.

La volta scorsa abbiamo visto che, per quanto comoda, l'interpolazione polinomiale non è affatto esente da "anomalie

comportamentali". Abbiamo bonariamente giustificato tale imprevedibile andamento come ribellione del polinomio alla crocefissione sui nodi. In termini meno sacrileghi possiamo affermare che un polinomio di n-esimo grado può avere n-1 gobbe e ventri, che spesso vanno a finire fra i nostri nodi causando quelle oscillazioni note come fenomeno di instabilità di Runge.

In base a questa analisi (semplicistica, ma non per questo sbagliata) possiamo concludere che un primo sistema per evitare, o quanto meno sminuire, l'instabilità dell'interpolazione polinomiale sia quello di ridurre il grado del polinomio.

in un senso che nell'altro (per gli amanti della formula 1 una cubica presenta una chicane) e c'è un punto a curvatura nulla.

Prima di proseguire ricordiamo che un polinomio di n-esimo grado è determinato dai valori di n+1 nodi.

Polinomi a tratti : introduzione

Abbiamo appena visto che è prudente non utilizzare polinomi di grado troppo elevato; capita però spesso di avere un numero di nodi molto maggiore di quello che ci consente il polinomio che vogliamo utilizzare. Pensiamo allora di ordinare i punti a disposizione secondo le x crescenti: in funzione del valore da estrapolare si scelgono dei nodi contigui in numero opportuno e quindi (col programma Interpola magari) si procede all'estrapolazione.

descritta più avanti, che implementa i metodi appena visti. Se la trattazione degli algoritmi non fosse stata abbastanza chiara consigliamo di provare il programma Interpola2 leggendo il paragrafo che lo descrive in calce all'articolo.

Consigliamo di predisporre un'uscita grafica per i vostri programmi, in modo particolare però questa volta, poiché i grafici possono consentire un'immediata comprensione del problema.

Un'altra buona idea è quella di salvare (e quindi poter caricare) i dati relativi ai nodi sul dischetto per evitare la noia di doverli scrivere ogni volta; un sistema più rapido, ma solo per fare delle prove, è quello di inserire delle funzioni che forniscano i nodi, funzioni che andranno richiamate ogni qual volta si definisce l'insieme dei dati.

Dall'interpolazione alla computer graphics usando i polinomi come pennelli

di Giovanni Michelon e Luigi Manzo

Premessa matematica

Nella speranza che i lettori superstiti alla prima puntata di questa serie superino anche lo choc del titolo di questo paragrafo, vediamo di introdurre alcuni concetti per favorire la comprensione degli algoritmi che seguiranno.

Facciamo intanto notare che spesso si confonderà la nozione di funzione con quella di grafico di una funzione, cosicché diremo retta sottointendendo un polinomio di primo grado, o parabola per uno di secondo o cubica per uno di terzo. Il riferimento al grafico permette però una visione geometrica, e perciò più familiare, del problema algebrico dell'interpolazione.

Consideriamo ora una curva (fig.1), preso un suo punto possiamo immaginare di trovare un cerchio di raggio opportuno che sia tangente alla curva in quel punto e vi si adatti meglio di ogni altro: il reciproco della misura del raggio si dice curvatura della curva nel punto. Per chiarire meglio il concetto di curvatura supponiamo che la curva sia una strada che stiamo percorrendo a bordo di un'automobile: giunti al punto prescelto blocchiamo lo sterzo, la macchina descriverà (burroni ai margini permettendolo) una circonferenza il cui raggio esprimerà la curvatura cercata. È ovvio che se una strada... una curva ha tutti i punti a curvatura nulla sarà rettilinea.

Presa una parabola notiamo che ha curvatura sempre dello stesso segno, si "sterza" cioè sempre nello stesso verso; una cubica invece presenta curvatura sia

Questo è un metodo senz'altro corretto ma piuttosto noioso da effettuare a mano più di una volta. Nulla ci vieta però di fare una scelta una volta per tutte e poi lasciare al calcolatore il compito di arrangiarsi! Ciò che ci proponiamo di fare è suddividere il nostro insieme ordinato di nodi in un certo numero m di sottoinsiemi contenenti ciascuno n+1 nodi: diciamo tratti ugnuno di questi sottoinsiemi; ogni tratto avrà in comune coi tratti adiacenti un nodo (vedi fig.2), il numero di nodi in totale sarà m*n+1. Pensiamo di numerare i nodi da 0 a m*n: il generico tratto k (con k da 1 ad m) sarà individuato dai nodi xn(n*(k-1)) e xn(n*k) e comprenderà i nodi fra questi.

Sul singolo tratto si procederà poi all'estrapolazione con un polinomio di grado (avrete senz'altro capito) n.

Assegnato dunque il numero totale dei nodi a disposizione e l'ordine del polinomio da utilizzare ci si calcolerà il numero di tratti, scartando, eventualmente, alcuni nodi (per esempio fra gli ultimi) nel caso che m non risulti un intero.

Suggerimenti

Per la realizzazione di vostri programmi potete usare le subroutine del programma Interpola visto nel numero precedente: ne proponiamo qui una seconda versione,

L'uso di una funzione predefinita può essere utile per un confronto con l'interpolante a tratti, potete così rendervi conto dell'errore che si commette con una scelta del grado del polinomio anziché con un'altra.

Commento ai risultati

Usando il metodo proposto potrete notare un comportamento come quello di fig.3. Dividendo in tratti l'insieme dei nodi abbiamo sì evitato l'instabilità legata ad ordini polinomiali troppo elevati e limitato perciò gli errori di approssimazione, ma ottenuto lo sgradevole effetto degli spigoli agli estremi di ogni tratto. Questi spigoli sono dovuti alla perdita di informazione sull'andamento globale della funzione descritta dall'insieme completo dei nodi: tale perdita di informazione è dovuta, purtroppo, proprio alla suddivisione in tratti.

Sembra proprio che ci siamo dati la zappa sui piedi! Prima di farci prendere dal più completo sconforto rianalizziamo i risultati ottenuti ed anche il problema stesso dell'interpolazione.

Innanzitutto, abbiamo un metodo stabile e possiamo perciò controllarne gli errori: ciò che ci disturba è lo spigolo ai bordi di ogni tratto, quello che geometricamente si vorrebbe ottenere è invece una curva quanto più "liscia" possibile. Con riferimento alla fig.4 possiamo imporre che il

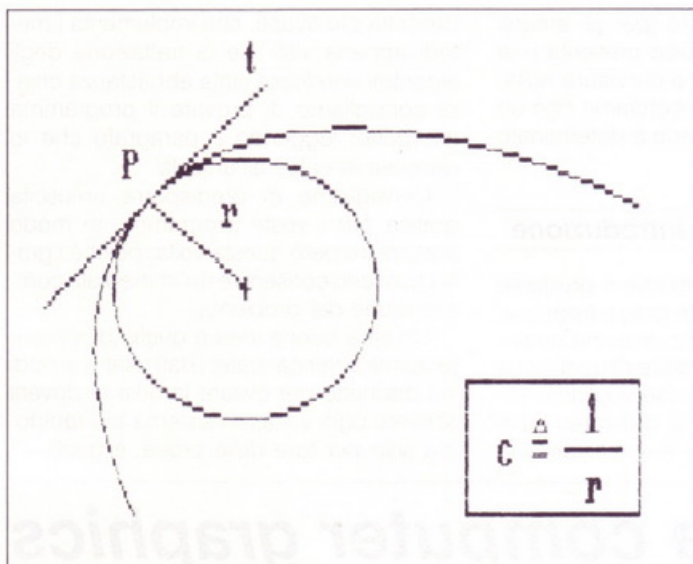



Fig. 1

Possiamo adottare tecniche migliori: anziché imporre una tangente prefissata agli estremi imponiamo che i polinomi su tratti adiacenti abbiano nei nodi di confine la stessa tangente (non una prefissata!).

Si notino le differenze fra i tre metodi fin qui accennati:

— il primo frantuma l'insieme dei nodi dimenticando ogni legame fra tratto e tratto;

— il secondo (Hermite) richiede altre informazioni, che ottiene in qualche modo, ma garantisce l'assenza di spigoli;

— il terzo elimina gli spigoli imponendo, attraverso le tangenti, un legame fra i tratti.

Gli ultimi due metodi si possono generalizzare: anziché richiedere solo che le tangenti siano uguali si può imporre che anche la curvatura sia la stessa. Per chia-

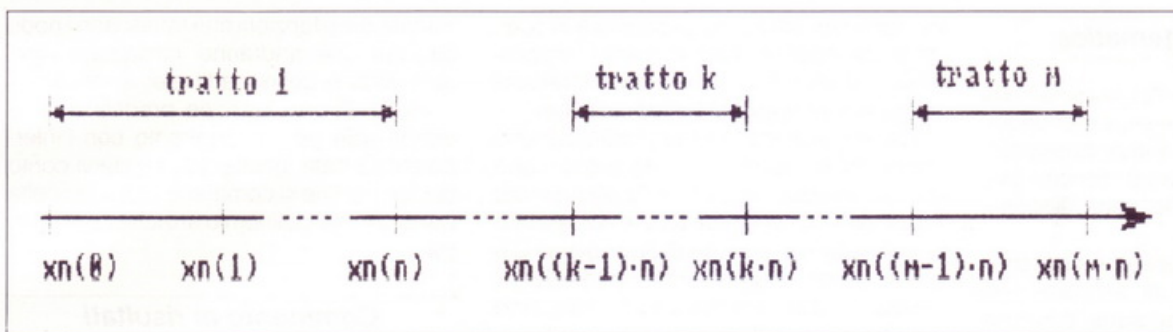
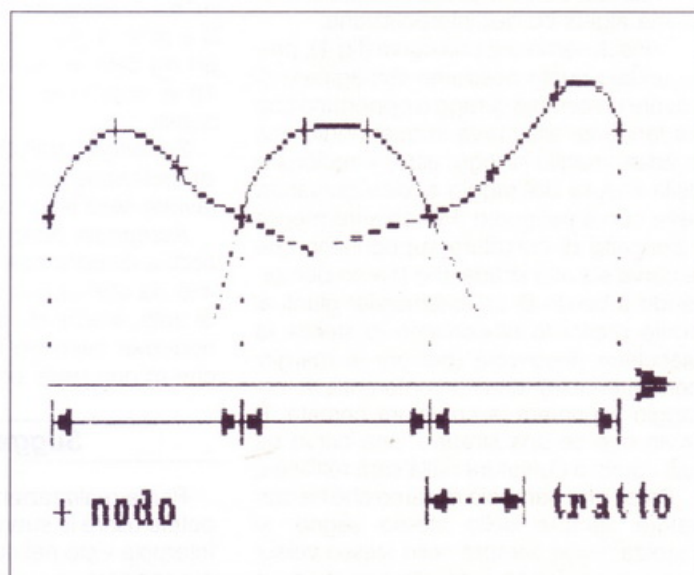



Fig. 2

polinomio interpolante sul generico tratto abbia una tangente prefissata sui nodi ai margini, tali ulteriori condizioni alzano di due il grado del polinomio (è questa la cosidetta interpolazione alla Hermite).

Con questo metodo otteniamo sì i vantaggi dell'interpolazione a tratti ed evitiamo gli spigoli, però dobbiamo fornire in più i dati relativi a tutte le tangenti nei nodi che sono estremo di tratto. Però potremmo non possedere tali informazioni (si pensi ai dati estratti da una tabella) oppure cercare in qualche modo di desumerle dai nodi adiacenti a quelli limite di tratto o magari fissarle a priori (il metodo basato su quest'ultima tecnica fissa le tangenti orrizzontali ed ottiene buoni risultati, si parla di metodo di Fejer). Noi non descriveremo tali metodi poiché richiedono una certa dimestichezza con la matematica che esula dalle pretese e dagli scopi di questo articolo.

Fig. 3

rire meglio questo discorso si osservi la fig.5, si ha un raccordo fra una retta ed un arco di circonferenza; nel punto di raccordo la tangente è ovviamente la stessa (coincide con la retta), la curvatura però è diversa subito prima e subito dopo il raccordo, si passa cioè da una curvatura nulla a quella data dal reciproco del raggio della circonferenza. Nel nostro gergo automobilistico è come se ci trovassimo a dover sterzare di colpo, anziché con gradualità, all'ingresso nella curva: potremmo dire che la curva in quel punto non è "liscia".

Siamo così giunti al variopinto mondo delle:

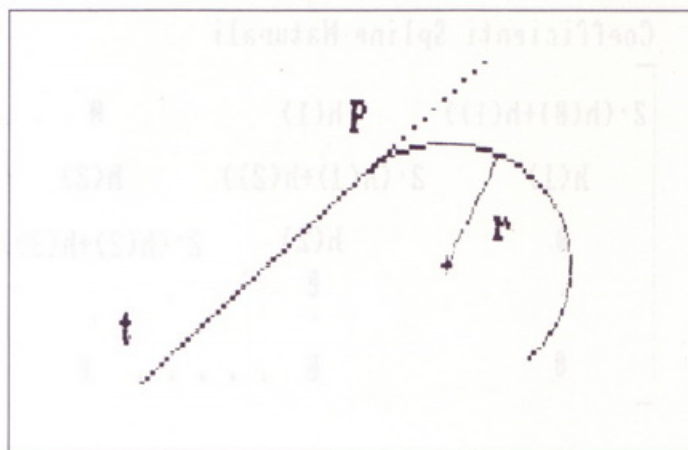



Fig. 4

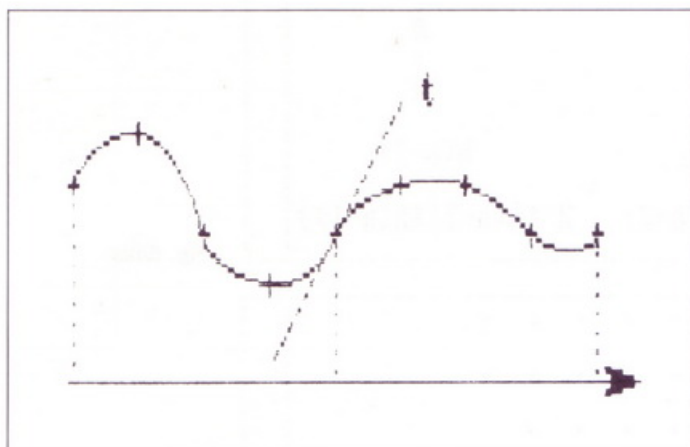



Fig. 5

Spline

Da quanto detto finora emerge già una descrizione delle spline: sono curve polinomiali a tratti con condizioni opportune agli estremi di ogni tratto tali da renderle "liscie" e da evitare che i singoli tratti risultino slegati.

Queste proprietà le rendono particolarmente simpatiche ed utili per la computer graphics poiché risolvono molto bene il problema del disegno di profili curvilinei.

In questo articolo discuteremo solo delle proprietà, del metodo di calcolo e dell'impiego delle spline: una trattazione teorica esauriente, anche se non complessa, ri-

1) $$s(t)=yn(k)+s1(k)\cdot(t-xn(k))+\frac{1}{2}s2(k)\cdot(t-xn(k))^2+\frac{1}{6\cdot h(k)}[s2(k+1)-s2(k)]\cdot(t-xn(k))^3$$

yn(k) :=(ordinata nel nodo k-esimo) $xn(k) \le t \le xn(k+1)$

s1(k),s2(k) :=(costanti da determinare)

h(k) :=(ampiezza tratto k-esimo) $h(k):=xn(k+1)-xn(k)$

2) $$s1(k)=\frac{yn(k+1)-yn(k)}{h(k)}-\frac{h(k)}{6}[2\cdot s2(k)+s2(k+1)]$$

Fig. 6

Coefficienti Spline Naturali

$$
\begin{bmatrix}
2\cdot(h(0)+h(1)) & h(1) & 0 & \cdots & 0 \\
h(1) & 2\cdot(h(1)+h(2)) & h(2) & & \vdots \\
0 & h(2) & 2\cdot(h(2)+h(3)) & \ddots & 0 \\
\vdots & 0 & \ddots & \ddots & h(n-2) \\
0 & 0 \cdots\cdots & 0 & h(n-2) & 2\cdot(h(n-2)+h(n-1))
\end{bmatrix}
$$

Coefficienti Spline Periodiche

$$
\begin{bmatrix}
2\cdot(h(0)+h(n-1)) & h(0) & 0 & \cdots & 0 & h(n-1) \\
h(0) & 2\cdot(h(0)+h(1)) & h(1) & & & 0 \\
0 & h(1) & 2\cdot(h(1)+h(2)) & \ddots & & 0 \\
\vdots & 0 & \ddots & \ddots & & h(n-2) \\
h(n-1) & 0 \cdots\cdots & 0 & h(n-2) & & 2\cdot(h(n-2)+h(n-1))
\end{bmatrix}
$$

Fig. 6/bis

Termini Noti Spline Naturali

s2(n)=s2(0)=0

{ n-1 termini da 1 a n-1 }

$$
\begin{bmatrix}
\cdots \\
\cdots \\
\cdots \\
6\left(\dfrac{yn(k+1)-yn(k)}{h(k)} - \dfrac{yn(k)-yn(k-1)}{h(k-1)}\right) \\
\cdots \\
\cdots
\end{bmatrix}
$$

per k da 1 a n-1

Termini Noti Spline Periodiche

s1(n)=s1(0)

s2(n)=s2(0)

{ n termini da 0 a n-1 }

$$
\begin{bmatrix}
6\left(\dfrac{yn(1)-yn(0)}{h(0)} - \dfrac{yn(n)-yn(n-1)}{h(n-1)}\right) \\
\cdots \\
6\left(\dfrac{yn(k+1)-yn(k)}{h(k)} - \dfrac{yn(k)-yn(k-1)}{h(k-1)}\right) \\
\cdots \\
\cdots
\end{bmatrix}
$$

per k da 1 a n-1

Fig. 6/tris

chiederebbe troppo spazio su queste pagine e strumenti matematici non immediatamente comprensibili, riportiamo allora in fine articolo un minimo di bibliografia per coloro che desiderassero una esposizione più tecnica.

Parleremo delle spline cubiche, cioè il polinomio interpolante sul singolo tratto sarà di terzo grado. Non sembri limitativa questa scelta, di fatto le spline cubiche sono le più usate sopratutto per la grafica al calcolatore; il terzo grado è anche il minimo possibile per una spline. Vediamo allora come è fatta: ogni tratto ha solo due nodi, cioè è composto dai soli nodi di estremo; su ogni tratto si ha un particolare polinomio di terzo grado che ha, come deve essere, le stesse tangenti e le stesse curvature dei polinomi dei tratti adiacenti nei due nodi. Rimangono però il primo e l'ultimo nodo dell'insieme di tutti i nodi; su questi due punti si assegnano valori a priori per la tangente e per la curvatura. Si usano due tipi di spline cubiche a seconda delle condizioni imposte su tali nodi:

— naturali: impongono che la curvatura sia nulla;

— periodiche: impongono stessa curvatura e stessa tangente.

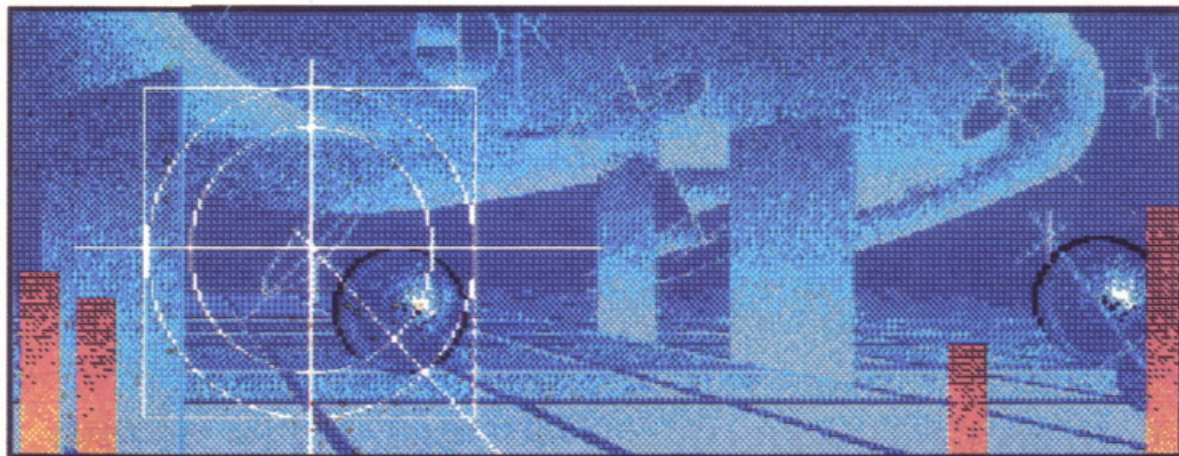
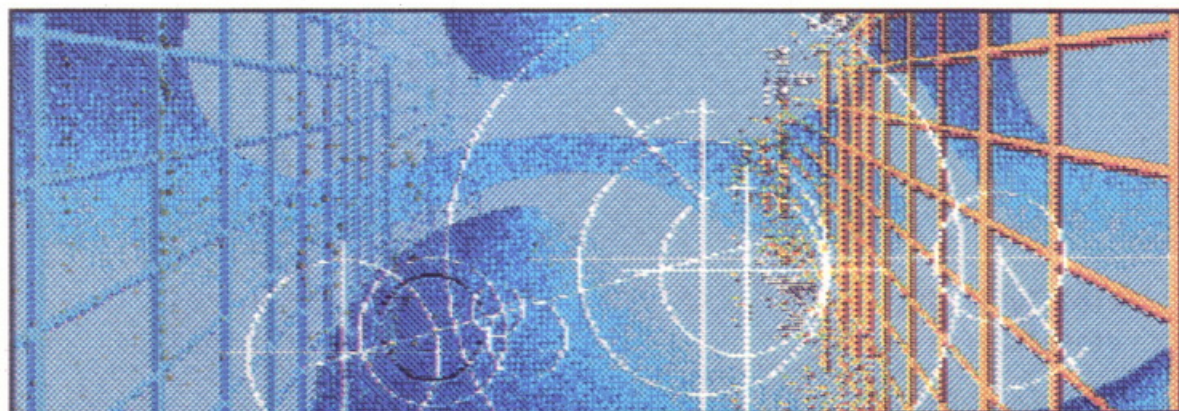
Tratteremo due diverse rappresentazioni delle spline cubiche, la prima più semplice e adatta all'interpolazione la seconda più complessa ma che meglio si adatta ai problemi di natura grafica.

Prima di procedere alla descrizione dei metodi di calcolo è opportuno far notare che i pregi delle spline si pagano con una maggior mole di conti, può però valerne la pena!

Spline per l'interpolazione

Supponiamo di aver numerato i nodi da 0 a nt (si noti che il numero dei tratti (nt) è uguale al numero dei nodi meno uno),

Fig. 7

Grafica di Amiga su Xerox

Grafica di Amiga su Xerox

numeriamo pure i tratti da 0 a nt-1, il polinomio sul generico tratto k sarà dato dalla formula 1 di fig.6 dove t assume i valori fra gli estremi del tratto, le altre costanti hanno il significato scritto in figura; le s1(k) si calcolano con la formula 2, le s2(k) si calcolano invece risolvendo uno dei due sistemi di equazioni lineari: uno per il caso naturale e l'altro per quello periodico. Di tali sistemi riportiamo solo le tabelle dei coefficienti e dei termini noti assumendo che le variabili siano nel primo caso s2(1) fino a s2(nt-1) con s2(0) = s2(nt) = 0 (curvatura nulla agli estremi) e nel secondo s2(0) fino a s2(nt-1) con s2(nt) = s2(0) e s1(nt) = s1(0) (stessa curvatura e stessa tangente agli estremi).

Per chi sa un po' di matematica, non sarà difficile riconoscere il polinomio di Taylor.

Il modo di procedere è allora il seguente:

— assegnati i nodi, si calcolano le ampiezze dei singoli tratti,

— con queste ed i valori dei nodi si costruisce il sistema,

— risoltolo si ottengono le costanti s2(k) (tenendo conto delle condizioni iniziali), con k da 0 ad nt,

— da queste si ricavano con la formula 2 le costanti s1(k), per k da 0 a nt.

Abbiamo così tutte le costanti necessarie per tutti gli nt polinomi di terzo grado nella variabile t.

La spline si traccia tratto per tratto o si calcola nel singolo punto dopo aver identificato il tratto cui appartiene.

B-spline

È questo il nome dato alle spline cubiche quando si passa ad una particolare rappresentazione che permette di trattare "curve piane generalmente regolari".

Per ottenere una qualunque di queste curve dal nome tanto lungo si procede nel seguente modo: ci si munisce di carta e penna e si traccia una qualunque curva senza spigoli venga in mente, senza sollevare mai la penna dal foglio; l'intersezione con archi di curva già tracciati è lasciata alla libera iniziativa del lettore.

Immaginiamo di fissare dei punti rappresentativi (che chiameremo nodi, tanto per cambiare) su questa curva man mano che la tracciamo; siano tali punti in numero di n numerati da 1 ad n: i punti coincidenti tracciati in momenti successivi si devono contare. Diremo che la curva è chiusa se il primo e l'ultimo nodo coincidono.

Se i punti sono stati scelti bene, allora la B-spline (o spline base) ripercorrerà la curva da voi tracciata.

Per una B-spline sono necessari, come per le spline viste prima, almeno due punti: il primo e l'ultimo. Anche qui si hanno due tipi di B-spline che potremmo chiamare ancora naturale e periodico poiché impongono agli estremi le medesime condizioni viste prima; in fig.7 sono riportate due B-spline sugli stessi quattro nodi (la curva è chiusa!) una naturale ed una periodica.

$\underline{x}(k) \quad k=1 \, . \, . \, n$

$\underline{y}(k) \quad k=0 \, . \, . \, n+1$

Fig. 8

Fig. 9

Formula di calcolo delle B-spline

$$\underline{x}_k(t) = \frac{1}{6}[\, t^3,\ t^2,\ t,\ 1\,]\cdot C \cdot \begin{bmatrix} \underline{v}(k-1) \\ \underline{v}(k) \\ \underline{v}(k+1) \\ \underline{v}(k+2) \end{bmatrix} \qquad C \doteq \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

$\underline{x}_k(t)$ punto generico sul tratto k

per k da 1 a n-1 con t : $0 \leq t \leq 1$ su ogni tratto

```
Calcolo costanti sul tratto

Ciclo su i da 1 a 4
   s←0
   Ciclo su j da 1 a 4
      s←s + C[i,j]·v(k+j-2)
   Fine j
   y(i)←s/6
Fine i
```

```
Calcolo generico punto x_k(t)

t0←t
s ← y(4)
Ciclo su i da 3 ad 1
   s ← t0·y(i)
   t0←t0·t
Fine i
x_k(t) ← s
```

Fig. 10

Il calcolo delle B-spline è leggermente più complicato di quello delle spline semplici ma.... ne vale la pena!

Per fissare le idee bisogna far notare che ora, occupandoci di una curva piana, non possiamo più ritenere, in generale, che la y sia funzione della x: abbiamo bisogno di un'ulteriore incognita t. Se il discorso non vi è chiaro, potete pensare che la t sia il tempo e x(t), y(t) siano le coordinate del punto tracciato dalla vostra penna all'istante t. Diremo parametrica tale rappresentazione della curva, il parametro è ovviamente t. Indicheremo (vedi fig.8) con x(k) il generico punto, la sottolineatura (in figura) serve a ricordare che x è un punto del piano ed ha quindi due coordinate: ascissa e ordinata. Per costruire la B-spline si devono trovare dei parametri v(k) (anche questi punti del piano); i v(k) sono in numero di n+2, numerati da 0 ad n+1, tali due punti in più servono per fissare le condizioni iniziali: spline naturale o periodica. Anche qui per trovare i v(k) si deve risolvere un sistema, uno per il caso naturale ed uno per il caso periodico. Riportiamo in fig.9 i coefficienti dei due sistemi, i termini noti sono i valori dei nodi x, le incognite sono v(1) fino a v(n), v(0) e v(n+1) si ottengono come riportato nella stessa figura, le somme ed i prodotti vanno intesi ascissa con ascissa ed ordinata con ordinata. In realtà ognuno di questi sistemi è un sistema per le ascisse ed uno per le ordinate di v ed x; poiché i coefficienti non cambiano è conveniente risolvere il sistema in parallelo per ascisse ed ordinate ottenendo così un buon guadagno in tempo di esecuzione.

Come prima le B-spline vanno tracciate fra un nodo e l'altro, in fig.10 è riportata la formula vettoriale per il calcolo del singolo punto in dipendenza dal parametro t, che varia fra 0 ed 1 su ogni tratto. Poiché la formula non è di immediata comprensione è riportato l'algoritmo di calcolo nella stessa fig.10. Prima di passare alla descizione dei programmi facciamo notare che la parte della formula di fig.10 che non dipende da t è costante sul singolo tratto, per cui conviene calcolarla prima di tracciare i punti del tratto.

Descrizione dei programmi

Interpola2 è la versione modificata di Interpola con l'introduzione delle spline. Come la versione precedente, permette di effettuare estrapolazioni, ma, cosa interessante dal punto di vista di chi si avvicina per la prima volta a problemi interpolatori,

permette di avere un confronto immediato fra la funzione interpolanda e quella interpolatrice attraverso i loro grafici.

Una volta lanciato, il programma chiede il numero dei tratti e dei nodi sul tratto, ricordandovi che questo è due se poi volete usare le spline, e il modo in cui volete inserire i nodi:

— l'inserimento manuale porta ad un ciclo in cui si richiedono i nodi da tastiera;

— l'inserimento da funzione porta a ricavare i nodi dalla funzione definita nelle prime linee del programma; vi verranno prima richiesti il primo e l'ultimo nodo, i rimanenti saranno ricavati in modo che i nodi risultino equidistanti.

Viene poi presentato un menù che vi permette di scegliere il metodo; a questo punto potete optare per un'uscita grafica o numerica, nel primo caso si richiedono gli estremi iniziali e finali per l'interpolazione (è buona norma che siano inclusi fra il primo e l'ultimo nodo), il passo di calcolo per il grafico ed un fattore di scala per le ordinate (tale fattore è valutato automaticamente per le ascisse); il passo di calcolo è l'incremento della variabile x; il fattore di scala richiesto si calcola dividendo per cento il massimo valore assoluto delle ordinate. Potete notare che i nodi che sono margini di tratto sono evidenziati con delle crocette.

Nel caso di uscita numerica si chiede il valore da estrapolare e quindi viene fornito il risultato; se non volete calcolare un altro valore premete il tasto u. Nota Bene: il programma prevede che si utilizzino solo le lettere minuscole.

Vediamo ora una breve descrizione del funzionamento di Interpola2: c'è un corpo principale che si occupa dell'ingresso dei dati e di coordinare i sottoprogrammi di calcolo. I sottoprogrammi NEWTON e NEVILLE sono quelli di Interpola: c'è solo la variabile b che serve a identificare il primo nodo di ogni tratto. Per quanto riguarda le spline c'è un primo modulo DISPONI che prepara la tabella dei coefficienti e dei termini noti (si faccia attenzione alle differenze indotte dalla variabile fspl) e, dopo aver chiamato un sottoprogramma che risolve il sistema e calcola le costanti s2, prepara anche le costanti s1. Terminato DISPONI si calcola il valore della spline scelta con il sottoprogramma SPLINE.

Il secondo programma si chiama B-spline e si pilota interamente col mouse; col tasto menù del mouse potete selezionare le seguenti opzioni:

— ClrScr = cancella lo schermo;

— New = fa ripartire il conteggio dei nodi;

— Aperta/Chiusa = indica il tipo di B-spline che verrà tracciata, aperta sta per naturale e chiusa per periodica; per scegliere una o l'altra di queste opzioni selezionate col tasto menù finché viene indicato il modo che desiderate;

— Trace = traccia la B-spline: è abilitato solo dopo che sono stati inseriti tre nodi;

— End = termina il programma.

Per fissare un nodo si usi il tasto sinistro, una crocetta segnalerà il nodo sullo schermo; i nodi vengono inseriti automaticamente in una tabella. Il programma stampa la posizione della freccetta e il numero dei nodi già inseriti.

Il programma è dimensionato per un massimo di ventinove nodi, ma dovrebbe esservi facile cambiare tale valore.

Per quanto riguarda il flusso del programma c'è un modulo principale che non fa quasi niente, aspetta infatti che voi "clickiate" qualcosa. La subroutine DISPONI prepara il menù discendente; le altre due subroutine SCELTA e NODO si occupano rispettivamente del tasto destro e sinistro del mouse. L'algoritmo di calcolo e tracciamento è contenuto nel modulo TRACE, il quale prima prepara le tabelle dei coefficienti, poi risolve i sistemi e quindi traccia la B-spline. Se non vi sono chiare le notazioni per il calcolo delle B-spline forse osservare il sottoprogramma PREPARA e TRACCIA potrà facilitare la vostra comprensione. Il lettore più smaliziato non dovrebbe avere grosse difficoltà a modificare e rendere più gradevole l'uscita grafica di B-spline (magari congiungendo i punti...).

Conclusioni

Sperando che dedichiate un po' di tempo ad Interpola2 invece che divertirvi solo con B-spline, vi facciamo notare che in tutto questo articolo si è parlato di sistemi lineari di equazioni (fate attenzione i sistemi visti qui e l'algoritmo, che li risolve, sono particolari!) senza spiegare come si risolvono con un calcolatore. Dipenderà dall'interesse suscitato la trattazione di questo argomento in un prossimo articolo.

Bibliografia per l'interpolazione a tratti e le spline

Josef Stoer
Introduzione all'analisi numerica Vol.1
edizioni Nicola Zanichelli Bologna 1974

Marco Cugiani
Metodi dell'analisi numerica
edizioni U.T.E.T. Torino 1972

N.B: su questi due libri non sono descritte le B-spline, non ci è noto un testo italiano, o tradotto, che ne parli. Il primo testo è piuttosto completo, richiede una buona conoscenza della matematica e la capacità di realizzare autonomamente gli algoritmi descritti. Il secondo, di più agevole lettura, si presta meglio per un primo approccio; purtroppo non descrive le spline. Questi due testi dovrebbero essere facilmente reperibili in una biblioteca.

```
'-----------------------------------------------------------------------
' PROGRAM interpola2                                  by G 1 M  1988
'-----------------------------------------------------------------------
DIM SHARED xn(50),yn(50)   ' Valori dei nodi
DIM SHARED Nw(50),P(50)    ' Vettori per Newton e Neville
DIM SHARED A(50,50),b(50)  ' Tabelle coefficienti, termini noti
DIM SHARED s1(50),s2(50)   ' e costanti per le spline
'-----------------------------------------------------------------------
DEF FN f(x)=SIN(x)
'-----------------------------------------------------------------------
' MAIN BODY
'-----------------------------------------------------------------------
 CLS
 fl=0
 LOCATE  4,20 : INPUT "Numero tratti                          ";NumTratti
 LOCATE  6,20 : INPUT "Numero nodi per tratto (Spline=2)      ";NumNodi
 LOCATE  8,20 : INPUT "Inserimento M)anuale F)unzione         ";Modeins$

 n=NumNodi-1 : m=NumTratti
 IF Modeins$="m" THEN
   lin=10
   FOR j=0 TO m*n
      LOCATE lin,20 : PRINT USING"Nodo ##";j
      LOCATE lin,40 : INPUT "x ";xn(j)
      LOCATE lin,50 : INPUT "y ";yn(j)
```

```
      lin=lin+1 : IF lin>23 THEN CLS : lin=4
   NEXT j
 END IF
 IF Modeins$="f" THEN
   LOCATE 10,20 : INPUT "Nodo inferiore      ";NInf
   LOCATE 12,20 : INPUT "Nodo superiore      ";NSup
   h=(NSup-NInf)/(m*n)
   FOR  j=0 TO m*n
      'xn(j)=NInf+h*j
      'yn(j)=FN f(xn(j))
   NEXT j
 END IF

 CLS
 LOCATE 12,20 : PRINT "a) Newton"
 LOCATE 14,20 : PRINT "b) Neville"
 LOCATE 16,20 : PRINT "c) Spline naturali"
 LOCATE 18,20 : PRINT "d) Spline Periodiche"
 LOCATE 20,20 : INPUT "scelta     (a-d)     ";met$
 CLS
 LOCATE  4,20 : INPUT "G)rafico  V)alore      ";ModeCalc$
 IF ModeCalc$="g" THEN
      LOCATE  8,20 : INPUT "Estremo inferiore             ";EInf
      LOCATE 10,20 : INPUT "Estremo superiore             ";ESup
      LOCATE 12,20 : INPUT "Passo grafico                 ";PGra
      LOCATE 14,20 : INPUT "Fattore di scala verticale ";fv
```

```
    GOSUB traccia
 END IF
 IF ModeCalc$="v" THEN
      c$=""
      WHILE c$<>"u"
        LOCATE  8,20 : INPUT "Valore x                    ";x
        LOCATE 12,20 : PRINT "Risultato                   ";
        GOSUB calcola
        INPUT "'u' per uscire ";c$

 END SUB 'crocetta
'---------------------------------------------------------------------
calcola:

 IF met$="a" OR met$="b" THEN n=NumNodi-1
 IF met$="c" THEN fspl=1 : n=NumTratti
 IF met$="d" THEN fspl=0 : n=NumTratti

 IF met$="a" OR met$="b" THEN              'newton  & neville
   tratto=1
   WHILE x<xn((tratto-1)*n)
     tratto=tratto+1
   WEND
   b=(tratto-1)*n
   IF met$="a" THEN
                 CALL newton(x)            'calcola newton
                 PRINT nwt
               ELSE
                 CALL neville(x)           'calcola neville
                 PRINT nev
   END IF
 END IF

 IF met$="c" OR met$="d" THEN              'calcola spline
   CALL disponi
   k=0
   WHILE x<xn(k)
     k=k+1
   WEND
   CALL spline(x,k)
   PRINT spl
 END IF
 RETURN 'calcola
'---------------------------------------------------------------------

SUB neville(x) STATIC
SHARED xn,yn,n,nev,b
 FOR i=0 TO n
   P(i)=yn(i+b)
 NEXT i
 FOR k=1 TO n
   FOR i=0 TO n-k
     P(i)=( (x-xn(i+k+b))*P(i)-(x-xn(i+b))*P(i+1) )/(
xn(i+b)-xn(i+k+b) )
   NEXT i
 NEXT k
 nev=P(0)
END SUB 'neville
'---------------------------------------------------------------------
SUB coeff STATIC
SHARED xn,yn,Nw,n,fl,b
 FOR i=0 TO n
   Nw(i)=yn(i+b)
 NEXT i
 FOR k=1 TO n
   FOR i=n TO k STEP -1
     Nw(i)=( Nw(i)-Nw(i-1) )/( xn(i+b)-xn(i-k+b) )
   NEXT i
 NEXT k
 fl=1              ' coefficienti calcolati

      WEND
 END IF

END 'interpola2
'---------------------------------------------------------------------
traccia:

 CLS
 IF met$="a" OR met$="b" THEN n=NumNodi-1
 IF met$="c" THEN fspl=1 : n=NumTratti
 IF met$="d" THEN fspl=0 : n=NumTratti
 fh=600/(ESup-EInf)

 IF met$="a" OR met$="b" THEN                      'newton & neville
   FOR tratto=1 TO NumTratti
     b=(tratto-1)*n
     CALL crocetta(b)
     FOR x=xn(b) TO xn(b+n) STEP PGra
       IF met$="a" THEN
                       CALL newton(x)              'traccia newton
                       CALL punto(x,nwt,1)
                    ELSE
                       CALL neville(x)             'traccia neville
                       CALL punto(x,nev,1)
       END IF
       IF Modeins$="f" THEN CALL punto(x,FN f(x),3)
     NEXT x
     fl=0        'ricalcola i coefficienti di newton
   NEXT tratto
   CALL crocetta(NumTratti)
 END IF

 IF met$="c" OR met$="d" THEN                      'traccia spline
   CALL disponi
   FOR k=0 TO n-1
     crocetta(k)
     FOR x=xn(k) TO xn(k+1) STEP PGra
       CALL spline(x,k)
       CALL punto(x,spl,1)
       IF Modeins$="f" THEN CALL punto(x,FN f(x),3)
     NEXT x
   NEXT k
   CALL crocetta(n)
 END IF
 RETURN 'traccia
'---------------------------------------------------------------------

 SUB punto(x,y,c%) STATIC
 SHARED EInf,fh,fv
   h=INT(10+(x-EInf)*fh)
   v=INT(100-y*fv)
   PSET (h,v),c%
 END SUB 'punto
'---------------------------------------------------------------------
 SUB crocetta(NodoTratto) STATIC
 SHARED xn,yn,EInf,fh,fv
   h=INT(10+(xn(NodoTratto)-EInf)*fh)
   v=INT(100-yn(NodoTratto)*fv)
   LINE (h-2,v)-(h+2,v),2
   LINE (h,v-2)-(h,v+2),2
END SUB 'coeff
'---------------------------------------------------------------------
SUB newton(x) STATIC
SHARED xn,Nw,n,fl,nwt,b
 IF fl=0 THEN CALL coeff
 s=Nw(n)
 FOR k=n-1 TO 0 STEP -1
   s=Nw(k)+(x-xn(k+b))*s
 NEXT k
 nwt=s
END SUB 'newton
'---------------------------------------------------------------------
SUB disponi STATIC
SHARED A,xn,yn,b,s1,s2,n,fspl

  FOR i=0 TO n             ' azzera la tabella dei coefficienti
    FOR j=0 TO n
      A(i,j)=0
    NEXT j
  NEXT i                   ' calcola la tabella (linee seguenti)

  h0=xn(1)-xn(0) : h1=xn(2)-xn(1)
  A(1,1)=2*(h0+h1) : A(1,2)=h1
  b(1)=6*( (yn(2)-yn(1))/h1-(yn(1)-yn(0))/h0 )
  h0=h1
  FOR i=2 TO n-2
      h1=xn(i+1)-xn(i)
      A(i,i-1)=h0 : A(i,i)=2*(h0+h1) : A(i,i+1)=h1
      b(i)=6*( (yn(i+1)-yn(i))/h1-(yn(i)-yn(i-1))/h0 )
      h0=h1
  NEXT i
  h1=xn(n)-xn(n-1)
  A(n-1,n-2)=h0 : A(n-1,n-1)=2*(h0+h1)
  b(n-1)=6*( (yn(n)-yn(n-1))/h1-(yn(n-1)-yn(n-2))/h0 )

  IF fspl=0 THEN     ' aggiunge coefficienti per le spline periodiche
     h0=xn(1)-xn(0) : h1=xn(n)-xn(n-1)
     A(0,0)=2*(h0+h1) : A(0,1)=h0 : A(0,n-1)=h1
     A(1,0)=h0 : A(n-1,0)=h1
     b(0)=6*( (yn(1)-yn(0))/h0-(yn(n)-yn(n-1))/h1 )
  END IF

  CALL risolvi                    ' risolve il sistema

  IF fspl=0 THEN                  ' spline periodiche
              s2(n)=s2(0)
            ELSE                  ' spline naturali
              s2(0)=0
              s2(n)=0
  END IF

  FOR i=0 TO n-1                  ' calcola le costanti s1
    hi=xn(i+1)-xn(i)
    s1(i)=( yn(i+1)-yn(i) )/hi-( 2*s2(i)+s2(i+1) )*hi/6
  NEXT i

END SUB 'disponi
'---------------------------------------------------------------------

SUB risolvi STATIC
SHARED A,b,s2,n                   ' risolve il sistema

  FOR k=fspl+1 TO n-1
    FOR i=k TO n-1
      FOR j=k TO n-1
        A(i,j)=A(i,j)-A(i,k-1)*A(k-1,j)/A(k-1,k-1)
      NEXT j
      b(i)=b(i)-A(i,k-1)*b(k-1)/A(k-1,k-1)
    NEXT i
  NEXT k
  s2(n-1)=b(n-1)/A(n-1,n-1)
  FOR i=n-2 TO fspl STEP -1
    s=b(i)
    FOR j=i+1 TO n-1
      s=s-s2(j)*A(i,j)
    NEXT j
    s2(i)=s/A(i,i)
  NEXT i

END SUB 'risolvi
'---------------------------------------------------------------------
SUB spline(x,k) STATIC
SHARED yn,xn,s2,s1,n,spl

  dx=x-xn(k)
  spl=( s2(k+1)-s2(k) )/6/( xn(k+1)-xn(k) )*dx
  spl=( s2(k)/2+spl)*dx
  spl=( s1(k)+spl)*dx
  spl=yn(k)+spl

END SUB 'spline
'---------------------------------------------------------------------
```

di Giorgio Dose

FINALMENTE UNO STANDARD PER IL BASIC!

All'inizio degli anni sessanta i linguaggi per computer erano poco conosciuti e comprensibili solo da coloro che li avevano studiati per anni. Con l'aumentare dell'interesse, soprattutto da parte degli studenti, verso la programmazione e l'uso dei computer, si avvertì la necessità di disporre di un linguaggio di programmazione di facile apprendimento e di semplice utilizzo. Dalle menti fervide di un gruppo di studenti americani e dei loro professori uscì uno dei primi linguaggi interattivi per computer: il BASIC, the Beginners All-purpose Symbolic Instruction Code.

Da allora il Basic ha fatto molta strada e si può senz'altro affermare che è diventato il linguaggio più popolare e che la sua immediatezza d'uso ha contribuito non poco alla diffusione dei computer. Con l'implementazione del linguaggio nella memoria ROM e con la possibilità di usare il registratore a cassette per la conservazione dei dati, la vendita dei microcomputer ha preso letteralmente il volo.

Un uso sempre maggiore di questo linguaggio sulle macchine più disparate portò inevitabilmente il Basic originale a subire notevoli cambiamenti. I progettisti di software creavano continuamente nuove funzioni per aumentarne la flessibilità e per sopperire alla sua innata lentezza. Ben presto i dialetti del Basic differirono sempre di più uno dall'altro e, nonostante che alcuni di essi, come il MicrosoftBASIC, siano diventati molto popolari, esiste ancora oggi una elevata incompatibilità tra i vari sistemi.

Un programma, ad esempio, scritto per un computer con una risoluzione grafica di 200x200 pixel non potrà mai funzionare su un computer con grafica ad alta risoluzione senza che per questo vengano apportate notevoli modifiche.

TrueBASIC

Visti i molti dialetti del Basic, l'American National Standards Institute nominò una commissione che potesse esaminare i pregi e i difetti di ognuno. Prendendo il meglio dei vari dialetti venne creato uno standard che avrebbe dovuto portare il Basic al state-of-art dei linguaggi per computer.

Uno dei membri della commissione era Tom Kurtz che, con l'aiuto di John Kemeny e altri studenti, aveva ideato il linguaggio Basic originale. Gli stessi progettisti, attenendosi alle norme dello standard ANSI, svilupparono anche il TrueBASIC. È questo un linguaggio ideale per l'Amiga perché pur permettendo vecchi costrutti, come i numeri di linea e i vari GOTO e GOSUB, possiede alcune delle più avanzate strutture come SELECT CASE, DO-LOOP, DO-WHILE e DO-UNTIL. Il TrueBASIC è trasportabile; ogni programma scritto con questo linguaggio sull'Amiga può funzionare su un Macintosh o IBM, ad esempio, senza alcuna modifica, compresa la parte grafica.

L'Editor

L'editor è di grandissimo aiuto nella scrittura dei programmi. Sia il TrueBASIC che l'AmigaBASIC lavorano a tutto schermo ma l'editor del TrueBASIC è più veloce e più facile da usare.

L'editor del TrueBASIC è ricco di svariati gadgets. Molto importante è la finestra dei messaggi d'errore che compare nella parte inferiore dello schermo ed ha le dimensioni di una linea di testo. La finestra è normalmente vuota e cambia colore quando viene visualizzato il messaggio d'errore. Contemporaneamente il cursore si posiziona alla linea causa dell'errore. La finestra può contenere fino a 5 messaggi ma può visualizzare solo uno di essi alla volta. È necessario clickare sulla finestra per far comparire il messaggio successivo.

L'editor dispone delle normali funzioni quali scroll bars e frecce, cut and paste, search and replace eccetera ma dispone anche di ulteriori comandi che necessitano di una piccola spiegazione.

Include Legge un file da disco e lo inserisce nel programma immediatamente dopo la linea corrente.

Keep Cancella tutte le linee tranne quelle del blocco specificato.

Edit Visualizza, nella finestra di editing, solo le linee del blocco desiderato.

Move to block Muove il cursore al blocco (subroutine) definito con un nome.

Compile Salva il programma in forma compilata su un disk file; questo riduce la lunghezza del programma ed aumenta la velocità di esecuzione.

L'editor ha in sè tutte le caratteristiche di un word-processor ed infatti alcuni dei file di testo presenti sul dischetto sono stati scritti usando l'editor.

Finestre

Il TrueBASIC si presenta con tre finestre: Command, Source e Output con le quali è facile scrivere, eseguire e correggere i programmi.

Source è la finestra nella quale viene scritto il programma; l'editor a pieno schermo ed i menu fanno parte di questa finestra.

Command è la finestra nella quale si inseriscono i comandi basic in modo immediato per rilevare lo stato del sistema o per le operazioni di debugging. Volendo conoscere, ad esempio, il valore della variabile x, si apre la suddetta finestra, si scrive "Print x" e quindi si preme <RETURN>. Il valore di x comparirà di seguito. Volete una lista dei file presenti sul dischetto? Aprite la finestra Command, scrivete "Files" e quindi date <RETURN>. Il TrueBASIC immediatamente vi mostrerà i file della directory corrente. Qualsiasi comando del TrueBASIC può venir digitato nella finestra Command.

La finestra Output è quella dove appare l'uscita del vostro programma. Normalmente questa finestra è chiusa finché non si esegue un programma in TrueBASIC. Allora essa viene aperta ed il risultato del programma, testo e grafica, appare sullo schermo. La finestra può venir aperta an-

che manualmente e mantenuta aperta mentre si edita un programma nella finestra Source.

I programmi in TrueBASIC possono gestire fino a dieci finestre aperte contemporaneamente sullo schermo. Queste finestre possono contenere testo o grafica od entrambi. Solo una finestra alla volta risulta attiva ma è il programma che decide in quale finestra lavorare e in che momento.

Programmare in TrueBASIC

Se siete abituati a programmare in AmigaBASIC o MicrosoftBASIC rimarrete sorpresi dalla facilità d'uso del TrueBASIC. Usando questo linguaggio il programma *SpaceZap* è stato scritto in un pomeriggio. Il programma gestisce suoni, grafica e imput da mouse in meno di 100 linee! Questo programma può girare su qualsiasi computer con il TrueBASIC mentre la versione con l'AmigaBASIC può funzionare solamente con l'Amiga. Per scrivere un programma equivalente in AmigaBASIC si sarebbe dovuto usare lo Sprite editor per creare uno sprite ed il numero delle linee sarebbe stato notevolmente maggiore.

Il TrueBASIC possiede molti comandi tradizionali del Basic ma ne comprende anche molti di inediti quali:

Il comando MAT, prefissato ad un comando BASIC, permette di manipolare i dati di un array senza creare un loop.

MAT READ - Opera come il tradizionale comando READ con la differenza che esso legge tutti gli elementi di un comando DATA e li pone in un array finché l'array stesso non è completo, e tutto questo senza ricorrere ad un ciclo FOR/NEXT. Ad esempio:

```
! Legge i mesi ed i giorni
! della settimana
!
dim mesi$(12), giorni$(7)
mat read mesi$, giorni$

data gennaio,febbraio,marzo,aprile,...
data lunedí, martedí, mercoledí ...
end
```

MAT PRINT visualizza tutti gli elementi di un array. Esempio:

```
! Stampa una piccola lista
!
dim a(10)

for i = 1 to 10         ! carica l'array
   let a(i) = i         ! con questo loop
next i
mat print a             ! stampa l'intero array
end
```

PICTURE è il comando che permette di definire una figura grafica e richiamarla come una subroutine. Picture può avere anche dei parametri in modo da rendere possibile delle variazioni ogni volta che il comamdo viene usato. Esempio:

```
! Figura di un poligono
!
picture poligono(lati)
   for i = 0 to lati              ! determina
      let u = 2*Pi*(i/lati)  ! i vertici
      plot Cos(u),Sin(u);  ! e traccia
   next i                           ! i lati
   plot
end picture
set window-1,1,-1,1 ! def. area disegno
for n = 3 to 10      !poligoni da 3-10 lati
  draw poligono(n)       ! disegna poligono
next n
end
```

TRANSFORMS è un comando per modificare completamente una figura prima di disegnarla. Il TrueBASIC prevede ben cinque funzioni per trasformare un disegno:

Shift(a,b) - muove la figura. Ogni punto (x,y) viene spostato al punto (x + a,y + b);

Scale(a) - varia la scala del disegno. Ogni punto (x,y) si ritrova in (x*a,y*a);

Scale(a,b) - come Scale(a) con la differenza che il punto disegnato in (x,y) va a finire in (x*a,y*b);

Rotate(a) - ruota la figura di (a) radianti (o gradi) in senso orario attorno alle coordinate di origine della finestra;

Shear(a) - inclina tutte le linee verticali del disegno in senso orario di (a) radianti o gradi. Il punto che si trova in (x,y) viene spostato alla locazione (x + y*tan(a),y).

La trasformazione può venir applicata anche agli array consentendo una facile manipolazione degli stessi.

ON-LINE HELP

Il TrueBASIC dispone di ulteriori facilitazioni nell'uso dell'help. Per richiamarlo basta digitare 'Help<CR>' nella finestra Command o premere il tasto help. Comparirà una richiesta; selezionare l'item per il quale servono le informazioni con un doppio click (o digitare le prime lettere del comando); si aprirà una finestra con il messaggio di help richiesto.

PLOT. Qualsiasi punto dello schermo può essere indirizzato usando il comando

'Plot x,y'

Volendo indirizzare piu' punti si userà

'Plot points: x1,y1;x2,y2;...'

Per disegnare linee tra i punti basta aggiungere il carattere ";" dopo le coordinate di ogni singolo punto

'Plot x1,y1;x2,y2;...'

Per riempire un'area delimitata da punti usare il comando

'Plot Area:x1,y1;x2,y2;...'

Il colore usato sara' quello corrente. Il comando Plot combinato con il comando Window consente di realizzare dei grafici molto velocemente e senza sforzo. Per la rappresentazione del grafico è necessario fornire innanzi tutto i valori di riferimento per gli assi cartesiani. Ad esempio, volendo disegnare un grafico che rappresenta il numero di automobili costruite in Italia, ogni cinque anni, negli ultimi trent'anni, si deve definire una finestra con il margine sinistro al valore 1958 ed il margine destro al valore 1988; il margine superiore potrà essere di 10 milioni mentre il margine inferiore sarà pari a 0. Una volta definita la finestra in questo modo si userà il comando plot, per l'inserimento dei punti significativi o il tracciamento delle linee, inserendo i valori delle coordinate x e y direttamente in anni e in milioni di automobili e non in pixel come normalmente siamo abituati a fare.

Il linguaggio TrueBASIC permette di far uso anche delle librerie. Questo significa

```
! SpaceZap                      !
! Implementazione in TrueBasic !
! di Amiga magazine            !
! Marzo 1988                   !
!                              !
!                              !
INPUT prompt "Livello di abilita' (0-5) ":skill    ! livello di abilita
LET skill=skill/5
SET back "black"                    ! sfondo al nero
SET window -500,500,-500,500        ! dimensiona schermo
LET ox=100                          ! posizione x iniziale della nave
LET oy=100                          ! posizione y iniziale della nave
LET a=9
LET shots=1
!
! disegna la nave
!
PLOT area:ox,oy+3*a;ox+2*a,oy-2*a;ox,oy;ox-2*a,oy-2*a;ox,oy+3*a
BOX KEEP ox-3*a,ox+3*a,oy+3*a,oy-3*a in ship$
                                    ! la forma della nave e posta in
CLEAR                               ! una variabile stringa
DO
   LET tx=0
   BOX SHOW ship$ at ox,oy          ! disegna nave
   BOX CLEAR ox,ox+6*a,oy+7*a,oy   ! cancella nave
   GET MOUSE: x,y,state    ! senti mouse per movimento nave
   IF abs(ox+3*a)-10<0 then LET tx=tx+1
   IF abs(oy+3*a)-10<0 then LET tx=tx+1    ! plot x e y
   IF state>0 then LET tx=tx+1      ! pulsante di sparo premuto?
                                    ! conteggio colpi
   IF state>0 then                  ! e suono dello sparo
      FOR ss=1 to 3
         SOUND 100,.03
         SOUND 500,.03
      NEXT ss
   END IF
   IF tx=3 then                     ! colpito ?
      LET score=score+1             ! se si, incrementa score
      LET ki=ki+1                   ! e incrementa contatore skill
      LET ex=rnd*50                 ! disegna esplosione
      SET color "red"
      FOR i=1 to ex
         IF i>(ex/2) then SET color "yellow"
```

```
         PLOT 0,0;rnd*100,rnd*100
         PLOT 0,0;-rnd*100,-rnd*100
         PLOT 0,0;-rnd*100,rnd*100
         PLOT 0,0;rnd*100,-rnd*100
      NEXT i
      CLEAR
      SET color "white"
      PRINT "Shots";shots            ! stampa colpi
      PRINT "Kills";score            ! stampa punteggio
      PRINT "Skill";skill*5          ! stampa livello
      LET ox=(rnd*1000)-500          ! pos. x per nuova nave
      LET oy=(rnd*1000)-500          ! pos. y per nuova nave
   END IF
   IF x1<>x then LET mx=x+x1         ! nave mancata, oalcola x
   IF y1<>y then let my=y+y1         ! nave mancata, calcola y
   LET ox=ox-mx/50
   LET oy=oy-my/50
   IF state>1 then
      GET MOUSE:N1,n2,null           ! reset pulsante di sparo
      SET color "red"                ! cambia colore puntatore
      LET shots=shots+1              ! aumenta di uno i colpi
   END IF
   PLOT 0,0              ! disegna puntatore nel colore corrente
   PLOT 50,0;100,0     ! bianco se non sparo
   PLOT -50,0;-100,0   ! rosso se sparo
   PLOT 0,50;0,100
   PLOT 0,-50;0,-100
   IF state>1 then SET color "white"  ! Sparo? cambia colore
   IF ki>4 then             ! se piu di 4 colpiti al livello scelto
      LET skill=skill+.2  ! incrementa skill level di 1
      LET ki=1
   END IF
   LET x1=x
   LET y1=y
   LET ox=ox+(skill*(rnd*35))   ! plot manovra evasiva nave
   LET oy=oy+(skill*(rnd*35))   ! in base al livello di abilita
   LET ox=ox+(skill*(-rnd*35))  ! piu alto il livello
   LET oy=oy+(skill*(-rnd*35))  ! piu si muove la nave
   IF oy>490 then LET oy=490
   IF oy<-490 then LET oy=-490
   IF ox>490 then LET ox=490
   IF ox<-490 then LET ox=-490
LOOP                                                        END
```

che se, ad esempio, avete una routine che usate frequentemente nei vostri programmi, potete sistemarla in una libreria e poi chiamarla da programma. (Queste librerie non hanno niente a che vedere con le librerie dell'Amiga, esse sono proprie del TrueBASIC). Verso e da le librerie possono essere passati valori e variabili; questo semplifica moltissimo la programmazione. Si possono scrivere e correggere diversi moduli indipendentemente uno dall'altro e quindi linkarli con un altro programma per produrre un programma completo.

Il package

Non rimarrete certo delusi nell'aprire la confezione del TrueBASIC. Troverete infatti: due dischetti, contenenti uno il programma principale e l'altro numerosi esempi di programmi, e due manuali: the User's Guide e the Reference Manual. Entrambi i manuali sono rilegati con spirale e pubblicati da Addison Wesley: lo stesso editore degli eccellenti Amiga reference manuals.

L'User's Guide è rivolto ai neofiti del Basic; il manuale infatti è una guida rapida per chi deve imparare a programmare con questo linguaggio; ogni comando è spiegato chiaramente ed è fornito di numerosi esempi.

Il Reference manual è invece rivolto ai programmatori già esperti che vi troveranno specifiche informazioni su ogni comando del TrueBASIC. Ogni capitolo del libro descrive nel dettaglio un diverso aspetto del linguaggio, dalla grafica al trattamento degli errori, ed è corredato di numerosi esempi. Nel manuale sono trattati concetti avanzati come l'assemblaggio di routine, la manipolazione grafica, il trattamento dei file e la compatibilità con i vari dialetti Basic.

In conclusione il TrueBASIC è veramente un programma accurato, di uso immediato ed esente da errori. Un semplice test consistente nel conteggio da uno ad un milione ha evidenziato una velocità almeno due volte superiore a quella dell'AmigaBasic.

Per il TrueBASIC sono disponibili un certo numero di librerie esterne di programmi. Tra queste ricordiamo 3D graphics, PC BASIC converter per tradurre da altri dialetti Basic, Communications Support, Sorting and Searching, Advanced String Library e diverse altre. L'Amiga Developers Toolkit comprende routine per grafica avanzata e gestione di suoni, routine per animazioni avanzate e accesso a tutte le funzioni di sistema.

All'articolo sono acclusi due semplici programmi per illustrare la semplicità d'uso e le possibilità offerte dal linguaggio.

SpaceZap

SpaceZap è stato scritto per valutare se il TrueBASIC supporta un buon livello di routine grafiche; con esso infatti non è possibile gestire gli sprite. È risultato comunque che il TrueBASIC dispone di una buona qualità grafica ed usando il comando Plot Area in unione con la funzione Picture è possibile simulare gli sprites come è stato fatto in questo programma.

Orbits

Orbits è un piccolo programma che era stato scritto originariamente per un computer Atari. Esso dimostra come sia facile convertire al TrueBASIC, programmi scritti in altri dialetti Basic e per altri computer.

Orbits permette di definire la gravità e la posizione di un massimo di dieci oggetti; il programma poi calcola e visualizza ogni oggetto in relazione a tutti gli altri.

Il programma è stato convertito abbastanza facilmente. Tutte le formule matematiche sono state trasferite intatte. Le uniche variazioni riguardano le dimensioni dello schermo e l'indirizzamento dello stesso con i comandi Window e Plot. Questi ultimi sono stati adattati per sfruttare il vantaggio di una maggiore risoluzione e di un numero più elevato di colori disponibili.

```
! Orbits                              !
! Implementazione in TrueBasic !
! di Amiga Magazine                   !
! Aprile 1988                         !
!                                     !
!                                     !
!                                     !
SET mode "high16"          ! grafica in alta risoluzione
SET window -500,500,-500,500
                           ! origine al centro dello schermo e
SET color "blue"           ! schermo di larghezza ed altezza di
SET back "black"           ! 1000 punti
DO while t<2 or t>10       ! numero degli oggetti
   PRINT "Numero degli oggetti orbitanti (massimo 10)";
   INPUT t
LOOP
!
! definizione array
!
DIM G(10),X(10),Y(10),U(10),V(10),R(10)
!
! valori per tutti gli oggetti
!
FOR i=1 to t
   PRINT "Per oggetto orbitante n. ";i  !valore gravita
   PRINT "Gravita=";
   INPUT G(i)
         let R(i)=G(i)/5
   PRINT "Coordinata X=";     ! coordinata X iniziale
   INPUT X(i)
   PRINT "Coordinata Y=";     ! coordinata Y iniziale
   INPUT Y(i)
   PRINT "X-velocita=";   ! valore velocita X
   INPUT U(i)
   LET U(i)=(U(i))/100
   PRINT "Y-velocita=";       ! valore velocita Y
   INPUT V(i)
     LET V(i)=(V(i))/100
     NEXT i
```

```
CLEAR
!
! calcola e definisci i punti
!
DO
   FOR i=1 to t
      FOR j=1 to t
         IF i<>j then
            LET X1=X(j)-X(i)
            LET Y1=Y(j)-Y(i)
            LET D2=X1*X1+Y1*Y1
            LET G1=G(j)/(D2*sqr(D2))
            LET U(i)=U(i)+G1*X1
            LET V(i)=V(i)+G1*Y1
         END IF
      NEXT j
      IF i=1 or i=9 then SET color "blue"
      IF i=2 or i=10 then SET color "white"
      IF i=3 then SET color "red"
      IF i=4 then SET color "yellow"
      IF i=5 then SET color "green"




      IF i=6 then SET color "cyan"
      IF i=7 then SET color "magenta"
      IF i=8 then SET color "brown"
      box circle X(i)-(R(i)/2),X(i)+(R(i)/2),Y(i)-
(R(i)/2),Y(i)+(R(i)/2)
      !PLOT X(i),Y(i);      ! plot prima posizione
      LET Y(i)=Y(i)+V(i)
      LET X(i)=X(i)+U(i)
      box circle X(i)-(R(i)/2),X(i)+(R(i)/2),Y(i)-
(R(i)/2),Y(i)+(R(i)/2)
      !PLOT X(i),Y(i)       ! plot seconda posizione
   NEXT i
! CLEAR                 ! rimuovere "!" per cancellare le tracce
LOOP
END
```

LISTINO LIBRI JACKSON

CODICE	TITOLO	PREZZO
	INFORMATICA: CONCETTI GENERALI	
511 A	COME PROGRAMMARE	15.000
503 A	PROGRAMMAZIONE STRUTTURATA, CORSO DI AUTOISTRUZIONE	15.000
101 H	TERMINI DELL'INFORMATICA E DELLE DISCIPLINE CONNESSE	50.000
539 A	LOGICA E DIAGRAMMI A BLOCCHI: TECNICHE DI PROGRAMMAZIONE	40.000
526 P	DATA BASE: CONCETTI E DISEGNO	22.500
GYS190	TRADUTTORI DI LINGUAGGI	26.000
G 240	PAROLE BASE DELL'INFORMATICA	8.000
GYS245	CONCETTI DI INFORMATICA	43.000
GYS248	DATA PROCESSING	45.000
GY 264	DATA FILE	50.000
GYS266	ARCHITETTURE DI SISTEMA	32.000
GY 354	SISTEMI INTELLIGENTI	28.000
CZ 419	ANALISI E PROGRAMMAZIONE	11.000
158 EC	INFORMATICA DI BASE I CONCETTI FONDAMENTALI HARDWARE E SOFTWARE	55.000
526 A	VOI E L'INFORMATICA	15.000
100 H	DIZIONARIO DI INFORMATICA	59.000
GY 551	I LINGUAGGI DELLA 4a GENERAZIONE	65.000
GYS552	PRIMA DEL LINGUAGGIO LA PROGRAMMAZIONE	35.000
GYS559	C.S.P. - PROCESSI SEQUENZIALI	49.000
GYS546	ALGORITMI FONDAMENTALI	54.000
GY 618	SISTEMI ESPERTI	28.000
047 T	MICROPROCESSORI	14.500
048 T	DATA BASE	14.500
049 T	FILE	14.500
CI 686	CAPIRE IL PERSONAL COMPUTER	35.000
G 540	MODELLI MATEMATICI E SIMULAZIONE	56.000
GE 688	ENCICLOPEDIA MONOGRAFICA DI ELETTR. E INF. VOLUME I	58.000
GE 689	ENCICLOPEDIA MONOGRAFICA DI ELETTR. E INF. VOLUME II	58.000
GY 629	SOFTWARE DI BASE - Strumenti di sviluppo	52.000
	INFORMATICA: SISTEMI OPERATIVI	
G 223	UNIX LA GRANDE GUIDA	70.000
GY 272	SISTEMI OPERATIVI PER MICROCOMPUTER	25.000
GY 273	MS-DOS LA GRANDE GUIDA	45.000
510 P	CP/M CON MP/M	29.000
CZ 538	MS DOS 2 E 3	49.000
G 543	XENIX	45.000
R 588	LAVORARE CON XENIX	70.000
GYS271	SISTEMI OPERATIVI	55.000
R 615	I COMANDI DI XENIX MAIL	12.500
092 D	SOFTWARE DI BASE E SISTEMI OPERATIVI	7.000
093 D	CP/M IL "SOFTWARE BUS"	7.000
094 D	MS-DOS E PC-DOS LO STANDARD IBM	7.000
009 H	UNIX	8.500
011 H	CP/M	8.500
044 T	MS DOS	14.500
045 T	PC DOS	14.500
R 628	MICROSOFT OS/2	50.000
046 T	UNIX	14.500
MS 02 E	COFANETTO "MS-DOS" 5"¼ - Corso autoistruzione	156.000 F
R 600	MS DOS ADVANCED - Il Manuale del Programmatore	55.000
GY 663	UNIX PHOGRAMMAZIONE AVANZATA	55.000
BY 724	GUIDA AI SISTEMI OPERATIVI	29.000
	INFORMATICA: LINGUAGGI	
501 A	IMPARIAMO IL PASCAL	16.000
502 A	INTRODUZIONE AL BASIC	25.000
500 P	PASCAL MANUALE E STANDARD DEL LINGUAGGIO	16.000
329 A	PROGRAMMARE IN ASSEMBLER	14.000
513 A	PROGRAMMARE IN BASIC	8.000
514 A	PROGRAMMARE IN PASCAL	19.000
516 A	INTRODUZIONE AL PASCAL	39.000
517 P	DAL FORTRAN IV AL FORTRAN 77 (II ED.)	32.000
521 A	50 ESERCIZI IN BASIC	17.000
525 A	BASIC PER TUTTI	23.000
534 A	MANUALE DEL BASIC	45.000
509 A	LOGO: POTENZA E SEMPLICITÀ	20.500

CODICE	TITOLO	PREZZO
507 B	TUO PRIMO PROGRAMMA IN BASIC (II)	19.500
533 A	BASIC DALLA A ALLA Z	19.000
540 A	LINGUAGGIO ADA	19.500
541 P	LINGUAGGIO C	25.000
542 P	COBOL STRUTTURATO: CORSO DI AUTOISTRUZIONE	50.000
508 P	PROGRAMMARE IN C	39.000
G 233	COBOL PER MICROCOMPUTER	35.000
GYS246	ESERCIZI DI FORTRAN	20.000
GYS247	ESERCIZI IN PASCAL: ANALISI DEI PROBLEMI	29.000
GYS254	PROGRAMMAZIONE IN LINGUAGGIO ADA	42.000
GY 270	APL PER IL P.C. IBM	25.000
GYS274	DAL PASCAL AL MODULA 2	26.000
GYS311	LINGUAGGIO C IL LIBRO DELLE SOLUZIONI	24.000
GYS328	APPLICAZIONI IN PASCAL	32.000
GY 535	TURBO PASCAL	29.000
G 544	"C" LIBRARY	49.000
GYS550	PROLOG - LINGUAGGIO E APPLICAZIONE	32.000
R 589	TURBOPASCAL - LIBRERIA DI PROGRAMMI	45.000
042 T	LINGUAGGIO C	12.500
108 D	FORTH ANATOMIA DI UN LINGUAGGIO	7.000
107 D	FORTRAN E COBOL LINGUAGGI SEMPRE VERDI	7.000
086 D	ED È SUBITO BASIC VOL. 1	7.000
087 D	ED È SUBITO BASIC VOL. 2	7.000
034 T	PROLOG	14.000
035 T	LISP	12.500
001 H	COBOL	8.500
006 H	PASCAL	8.500
007 H	BASIC	8.500
010 H	FORTRAN 77	8.500
020 H	LOGO	8.500
022 H	FORTH	8.500
R 612	TURBO PROLOG	50.000
GY 626	IL MANUALE DEL PASCAL	42.000
GY 616	DEBUGGING C	55.000
GY 687	DALLA PROGRAMMAZIONE STRUTTURATA AL PASCAL	42.000
GY 634	FONDAMENTI DI COMMON LISP	40.000
	INFORMATICA: LAVORO È SOCIETÀ	
519 P	COMPUTER GRAFICA	29.000
800 P	ODISSEA INFORMATICA	50.000
407 H	APPLICAZIONI DEL COMPUTER NELL'UFFICIO MODERNO	23.000
802 H	INFORMATICA MUSICALE	27.000
802 P	COMPUTERGRAPHIA	40.000
806 P	COMPUTER PER L'INGEGNERIA EDILE	22.000
807 P	COMPUTER PER IL MEDICO	19.000
CI 231	COMPUTER IMAGE	40.000
CI 241	ODISSEA INFORMATICA STRATEGIE CULTURALI PER UNA SOCIETÀ INF.	32.000
G 400	COMPUTER GRAPHICS E ARCHITETTURA	27.000
PV 409	COMPUTER GRAPHICS E MEDICINA	18.000
GY 487	MEDICO & COMPUTER	45.000
GY 548	INFORMATICA MEDICA	65.000
PA 685	OFFICE AUTOMATION	28.000
RA 596	DESKTOP PUBLISHING	35.000
050 T	WORD	14.500
	INFORMATICA: SOFTWARE PACCHETTI APPLICATIVI	
570 P	CONTABILITÀ COL PERSONAL COMPUTER	27.000
525 P	WORDSTAR	24.000
546 P	MANUALE DEL DBASE II	24.000
578 P	PC NELL'ORG. DELLE PICCOLE AZIENDE: APPL. DEL MULTIPLAN	29.000 F
PP 219	LOTUS 1-2-3: GUIDA ITALIANA ALL'USO	21.000
G 234	RIORDINO E GESTIONE DEGLI ARCHIVI APPLICAZIONI CON PFS-FILE	30.000
PP 255	DBASE III GUIDA ITALIANA ALL'USO	45.000
PA 282	MODELLI DECISIONALI PER IL MANAGER	50.000
PA 288	PIANIFICAZIONE AZIENDALE PLANNING, MARKETING STRAT., BUDGETING	35.000 F
PP 310	LA GRANDE GUIDA LOTUS A SYMPHONY	70.000
PP 326	MULTIPLAN CORSO D'ISTRUZIONE	40.000 F
PP 344	FRAMEWORK II - GUIDA ITALIANA ALL'USO	27.000
PP 351	WORD PROCESSING	27.000

CODICE	TITOLO	PREZZO
PP 467	IMPARA 1-2-3 CON LA GRANDE GUIDA LOTUS	45.000
PP 468	CHART - CORSO ISTRUZIONE	45.000 F
PP 473	IL NUOVO 1-2-3 GUIDA ALL'USO DELLA VERSIONE ITALIANA 2 LOTUS 1-2-3	29.000
PA 474	BILANCIO, BUDGET, CASH FLOW	40.000 F
PP 475	DBASE III - CORSO DI PROGRAMMAZIONE	23.000
PA 476	PREVISIONE, PIANIFICAZIONE, SIMULAZIONE CON LOTUS 1-2-3	60.000 F
PV 477	GUIDA ALLA BUSINESS GRAPHIC	20.000
PP 480	AUTOCAD	40.000
PP 481	RBASE 5000 - GUIDA ITALIANA ALL'USO	20.000
PP 537	IL MANUALE DI WINDOWS	60.000
PP 539	DBASE III - TECNICHE AVANZATE DI PROGRAMMAZIONE	42.000
PP 545	APPLICAZIONI DI DBASE III	50.000 F
PA 566	MODELLI DECISIONALI CON LOTUS 1-2-3	40.000 F
PP 577	MANUALE DBASE III PLUS	49.000
039 T	WORDSTAR	12.500
040 T	LOTUS 1-2-3	12.500
043 T	WINDOWS	12.500
PP 621	I COMANDI DI DBASE III PLUS	12.500
095 D	GUIDA AI PACKAGE APPLICATIVI MERCEOLOGIA DEL SOFTWARE	7.000
096 D	VISICALC GUIDA RAPIDA ALL'UTILIZZO	7.000
098 D	WORD PROCESSING	7.000
103 D	LOTUS 1-2-3 E SIMPHONY IL FASCINO DELL'INTEGRAZIONE	7.000
104 D	DBASE II E III I PRINCIPI DI DATABASE	7.000
106 D	MULTIPLAN SPREADSHEET MULTISTRATO	7.000
110 D	PACKAGE A CONFRONTO PROVE DEI SOFTWARE PIÙ DIFFUSI	7.000
031 T	FRAMEWORK E FRAMEWORK II	12.500
033 T	MULTIPLAN 2.02	12.500
036 T	SYMPHONY	12.500
038 T	REFLEX	12.500
027 H	EASY SCRIPT	8.500
033 H	PAGE MAKER	8.500
034 H	PROJECT	8.500
035 H	RBASE	8.500
PP 611	GUIDA ALL'USO PROFESSIONALE REFLEX	55.000
PP 636	MANUALE DI WORD	70.000
PP 594	GUIDA ALL'USO PROFESSIONALE DI LOTUS 1-2-3	50.000
PP 593	VENTURA - Il grande manuale	55.000
R 671	LINGUAGGIO C - Reference guide	12.500
051 T	I COMANDI DI LOTUS 1-2-3 - Reference guide	12.500
PP 581	PROGRAMMARE IN FRED	40.000
PP 631	dBASE III PLUS - Guida uso professionale	65.000
PP 694	PROGRAMMARE IN WINDOWS	70.000
PA 592	GESTIONE DELLA PRODUZIONE	40.000 F
PP 727	VENTURA - REFERENCE GUIDE	14.500
PP 700	MATEMATICA CON LOTUS 1-2-3	35.000 F
R 574	MANUALE DELLE STAMPANTI LASER	25.000
	PERSONAL COMPUTER	
550 D	PROGRAMMI PRATICI IN BASIC	15.000
515 H	BASIC E LA GESTIONE DEI FILE VOL. I: METODI PRATICI	15.000
551 D	75 PROGRAMMI IN BASIC PER IL VOSTRO COMPUTER	12.000
552 D	PROGRAMMI DI MATEMATICA E STATISTICA IN BASIC	20.000
554 P	PROGRAMMI SCIENTIFICI IN PASCAL	29.000
516 H	BASIC E LA GESTIONE DEI FILE - VOL. 2	17.000
CH 182	COMPUTER HARDWARE REALIZZ. PRATICHE PER GLI HC PIÙ DIFFUSI	18.000
CI 187	COMPUTER L'HOBBY E IL LAVORO	12.000
G 235	GRAFICA PER PERSONAL COMPUTER	39.000
GE 263	METODI DI INTERFACC. PERIFERICHE	43.000
GE 402	CORSO DI AUTOISTRUZIONE PER MICROCOMPUTER VOL. 1 + VOL. 2	35.000
PA 406	COME GESTIRE LA PICCOLA AZIENDA CON IL P.C.	22.000
PP 408	BUSINESS IN BASIC	23.000
CI 412	IL COMPUTER È UNA COSA SEMPLICE	15.000
CC 415	CONTROLLO DEI DISPOSITIVI DOMESTICI CON IL P.C.	23.000
159 GC	PERSONAL COMPUTER DAL SOFTWARE DI BASE ALLE APPLICAZIONI D'UFFICIO	55.000
R 587	HARD DISK - LA GRANDE GUIDA	75.000
084 D	INTRODUZIONE AI PERSONAL COMPUTER VIVERE COL PC	7.000
099 D	SCRIVERE UN'AVVENTURA, 1000 AVVENTURE COL PROPRIO PC	7.000
100 D	GRAFICA E BASIC LE BASI DELLA COMPUTERGRAFICA	7.000

CODICE	TITOLO	PREZZO	
085 D	HARDWARE DI UN PERSONAL COMPUTER DENTRO E FUORI LA SCATOLA	7.000	
101 D	GESTIONE DEI FILE IN BASIC E PASCAL VOL. 1	7.000	
102 D	GESTIONE DEI FILE IN BASIC E PASCAL VOL. 2	7.000	
113 D	DISEGNARE COL PERSONAL COMPUTER	7.000	
105 D	PERSONAL E HOME COMPUTER A CONFRONTO	7.000	
112 D	SUONO E MUSICA COL PERSONAL COMPUTER	7.000	
109 D	COSTRUIRSI UN PERSONAL DATABASE	7.000	
097 D	GUIDA ALL'ACQUISTO DI UN PERSONAL COMPUTER	7.000	
088 D	TO DO OR NOT TO DO COME AVER CURA DEL PROPRIO PC	7.000	
089 D	SOFTWARE STRUTTURATO CON ELEMENTI DI PASCAL	7.000	
090 D	DIZIONARIO DI INFORMATICA	7.000	
091 D	BASI DELLA PROGRAMMAZIONE STENDERE UN PROG. COME SI DEVE	7.000	
004 H	PROGRAMMAZIONE	8.500	
015 H	PROGRAMMI DI STATISTICA	8.500	

PERSONAL COMPUTER: COMMODORE

CODICE	TITOLO	PREZZO	
347 D	VOI E IL VOSTRO COMMODORE 64	24.000	
348 D	COMMODORE 64 - IL BASIC	28.000	
400 D	FACILE GUIDA AL COMMODORE 64	13.500	
400 B	COMMODORE 64 - FILE	19.000	
409 B	COMMODORE 64 - LA GRAFICA E IL SUONO	34.000	C
570 D	MATEMATICA E COMMODORE 64	26.500	C
350 D	LIBRO DEI GIOCHI DEL COMMODORE 64	24.000	C
575 D	TECNICHE DI PROGRAMMAZIONE SUL COMMODORE 64	16.500	
572 D	LINGUAGGIO MACCHINA DEL COMMODORE 64	35.000	F
576 D	SISTEMA TOTOMAC: LA NUOVA FRONTIERA DEL TOTOCALCIO	29.000	C
548 B	64 PERSONAL COMPUTER E C64	45.000	
SDP222	STATISTICA AD UNA DIMENSIONE CON IL C64	24.000	
CC 260	AVVENTURE (COMMODORE 64)	20.000	C
CC 320	AMIGA HANDBOOK	35.000	
CC 322	COMMODORE 128 OLTRE IL MANUALE	29.000	
CC 323	PROGRAMMI PER COMMODORE 128	29.000	C
CZ 541	128 E 64 - LE PERIFERICHE	32.000	
CC 564	MANUALE RIPARAZIONE C64	55.000	
CZ 532	MANUALE DI AMIGA	39.000	
002 H	COMMODORE 64	8.500	
CC 658	GRAFICA E SUONO PER C64 - 64PC - C128	35.000	F
CC 657	MANUALE DEL COMMODORE C64 - C64PC - C128	35.000	F
CC 627	AMIGA 500	55.000	
CC 750	C.128 LA GRANDE GUIDA	50.000	
CC 749	C.64 LA GRANDE GUIDA	50.000	

PERSONAL COMPUTER: IBM

CODICE	TITOLO	PREZZO	
564 D	PROGRAMMI UTILI PER IBM PC	19.000	
G 217	GRAFICA PER IL PERSONAL COMPUTER IBM	39.000	
GY 319	PC IBM MANUALE DEL LINGUAGGIO MACCHINA	45.000	
GY 335	MAPPING PC IBM GESTIONE DELLA MEMORIA	42.000	
PP 407	MANUALE BASE DEL PC IBM	22.000	
041 T	PC IBM	12.500	
R 609	SOLUZIONI AVANZATE PER IL PROGRAMMATORE	60.000	
CZ 751	AVVENTURE PER MS-DOS	35.000	F

PERSONAL COMPUTER: OLIVETTI

CODICE	TITOLO	PREZZO	
401 P	PRIMO LIBRO PER M24: MS DOS E GW BASIC	28.000	
401 B	OLIVETTI M10: GUIDA ALL'USO	18.000	
CL 216	BASIC IN 30 ORE PER M24 ED M20	32.000	
CZ 483	MANUALE OLIVETTI M19	42.000	
CZ 536	MANUALE PC 128 OLIVETTI PRODEST	29.000	
CZ 582	PROGR. PER PC 128 OLIVETTI PRODEST (CASS.)	27.000	

PERSONAL COMPUTER: MSX

CODICE	TITOLO	PREZZO	
CZ 181	30 PROGRAMMI PER MSX	20.000	C
417 D	MSX: IL BASIC	23.000	
CC 261	AVVENTURE (MSX)	20.000	C
CC 289	SUPER PROGRAMMI PER MSX	35.000	C
CC 336	MSX LA GRAFICA	25.000	
111 D	STANDARD MSX	7.000	

PERSONAL COMPUTER: APPLE

CODICE	TITOLO	PREZZO	
331 P	APPLE II GUIDA ALL'USO	31.000	
416 P	MACINTOSH NEGLI AFFARI: MULTIPLAN E CHART	16.500	
424 P	UN MAC PER AMICO: USO, APPLICAZIONI E PROGRAMMI PER MACINTOSH	12.000	
PP 224	MACINTOSH ARTISTA: MACPAINT E MACDRAW	16.000	
CCP277	APPLE IIC GUIDA ALL'USO	45.000	
CC 312	PROGRAMMI PER APPLE IIC	13.000	
CC 417	PROGRAMMI COMM. E FINANZIARI CON APPLE	22.000	
340 H	APPLE MEMO	15.000	
CC 576	IL MANUALE DELL'APPLE II GS	28.000	
003 H	APPLE IIE IIC	8.500	
CC 665	MICROSOFT BASIC PER APPLE MACINTOSH	32.000	F

PERSONAL COMPUTER: ATARI - AMSTRAD - SHARP

CODICE	TITOLO	PREZZO	
540 H	BASIC ATARI	18.000	
CC 330	PROGRAMMI PER AMSTRAD CPC 464 CPC 664 - CPC 6128	29.000	C
CC 331	PROGRAMMI PER ATARI 130XE	19.000	C
CC 471	MANUALE ATARI 520 ST E 1040 ST	28.000	
CC 486	WORD PROCESSING CON AMSTRAD PCW 8256/8512	35.000	
032 T	AMSTRAD PCW 8256 e PCW 8512	14.000	
028 H	AMSTRAD 464 E 664	8.500	

COMMUNICATION E TELEMATICA

CODICE	TITOLO	PREZZO	
309 A	PRINCIPI E TECNICHE DI ELABORAZIONE DATI	20.000	
518 D	TELEMATICA	28.000	
528 P	TRASMISSIONE DATI	27.000	
617 P	RETI DATI: CARATTERISTICHE, PROGETTO E SERVIZI TELEMATICI	40.000	
GYS314	ELABORAZIONE DIGITALE DEI SEGNALI: TEORIA E PRATICA	25.000	
PA 327	BANCHE DATI RICERCA ONLINE	26.000	
158 LC	COMUNICAZIONI DALLE ONDE ELETTROMAGNETICHE ALLA TELEMATICA	55.000	
CC 472	MODEM E PC USO E APPLICAZIONI	25.000	
GTS478	RETI LOCALI	44.000	
GTS479	IL MODEM - TEORIA, FUNZIONAMENTO	28.000	
R 542	TRASMISSIONE DATI PER PC	31.000	
GT 555	LA TELEMATICA NELL'UFFICIO	35.000	
R 601	COLLEGAMENTO TRA MICRO E MAINFRAME	39.000	
BT 655	MANUALE DI TV E VIDEO COMMUNICATION	45.000	

ELETTRONICA DI BASE E TECNOLOGIA

CODICE	TITOLO	PREZZO	
201 A	CORSO DI ELETTRONICA FONDAMENTALE CON ESPERIMENTI	35.000	
204 A	ELETTRONICA INTEGRATA DIGITALE	50.000	
205 A	MANUALE PRATICO DI PROGETTAZIONE ELETTRONICA	35.000	
200 A	SISTEMI DIGITALI: MANUTENZIONE, RICERCA ED ELIMINAZIONE GUASTI	28.500	
GES262	TECNOLOGIE VLSI	70.000	
GES390	ELETTRONICA INTEGRATA DIGITALE IL LIBRO DELLE SOLUZIONI	17.000	
CE 411	LA FISICA DEI SEMICONDUTTORI	10.000	
158 PC	ELETTRONICA DI BASE I FONDAMENTI DELL'ELETTRONICA ANALOGICA	55.000	
158 CC	ELETTRONICA DIGITALE VOL. 1 DALLE PORTE LOGICHE AI CIRCUITI INTEGRATI	55.000	
158 DC	ELETTRONICA DIGITALE VOL. 2 DAI BUS AI GATE ARRAY	55.000	
158 GC	ELETTROTECNICA ELETTROSTATICA ELETTROMAGNETISMO RETI ELETTR.	55.000	

ELETTRONICA APPLICATA

CODICE	TITOLO	PREZZO	
601 B	TIMER 555	10.000	
203 A	INTRODUZIONE AI CIRCUITI INTEGRATI DIGITALI	10.000	
612 P	MANUALE DEGLI SCR VOL. 1	28.000	
613 P	MANUALE DI OPTOELETTRONICA	15.000	
614 A	FIBRE OTTICHE	15.000	
GE 403	JFET MOS E DATA BOOK	20.000	
GE 404	TRANSISTOR DATA BOOK	32.000	
GE 405	METODI DI PROTEZIONE CONTRO LE SOVRATENSIONI	17.000	
CE 413	IL MANUALE DEGLI SCR E TRIAC	15.000	
CE 421	MANUALE DEI FILTRI ATTIVI	29.000	
CE 423	MANUALE DEI PLL PROGETTAZIONE DEI CIRCUITI	29.000	
CE 425	MANUALE DEGLI AMPLIFICATORI OPERAZIONALI	29.000	
CE 429	250 PROGETTI CON GLI AMPLIFICATORI DI NORTON	39.000	
CE 431	MANUALE DEI CMOS	25.000	
CE 485	IL COLLAUDO DELLE SCHEDE	18.000	
BE 557	I TRASDUTTORI	43.000	
BT 585	FIBRE OTTICHE	29.000	
BE 578	MANUALE DI ELETTRONICA	29.000	
BE 558	IL MANUALE DEL TECNICO ELETTRONICO	51.000	
BE 610	GUIDA ALLA STRUMENTAZIONE ELETTRONICA	34.000	
BE 619	MULTIMETRI DIGITALI	42.000	
BE 639	ENCICLOPEDIA DEI CIRCUITI INTEGRATI	60.000	
BE 654	MANUALE DI ELETTRONICA DEL COMPUTER	20.000	
701 P	MANUALE PRATICO DEL RIPARATORE RADIO TV	29.000	
705 P	IMPIEGO PRATICO DELL'OSCILLOSCOPIO	17.500	
615 P	PROGETTAZIONE DI SISTEMI DI ALTOPARLANTI	21.000	
CE 427	L'ELETTRONICA A STATO SOLIDO	25.000	
BE 718	77 SCHEDE PER IL RIPARATORE TV	40.000	
BE 723	MISURE DEI CIRCUITI ELETTRONICI	26.000	

ELETTRONICA: MICROPROCESSORI

CODICE	TITOLO	PREZZO	
310 P	NANOBOOK Z80 VOL. 1	20.000	
007 A	BUGBOOK VII	17.000	
314 P	TECNICHE DI INTERFACCIAMENTO DEI MICROPROCESSORI	31.000	
312 P	NANOBOOK Z80 VOL. III	25.000	
320 P	MICROPROCESSORI DAI CHIPS AI SISTEMI	29.000	
324 P	PROGRAMMAZIONE DELLO Z80 E PROGETTAZIONE LOGICA	21.500	
326 P	Z80 PROGRAMMAZIONE IN LINGUAGGIO ASSEMBLY	50.000	
328 D	PROGRAMMAZIONE DELLO Z80	40.000	
504 B	APPLICAZIONI DEL 6502	17.000	
503 B	PROGRAMMAZIONE DEL 6502	35.000	
505 B	GIOCHI CON IL 6502	19.500	
G 220	8086-8088 PROGRAMMAZIONE	40.000	
GY 265	ASSEMBLER PER IL 68000	70.000	
CE 410	IMPIEGO DELLO Z80	23.000	
158 HC	MICROPROCESSORI ARCHIT. PROGR. E INTERFAC. DEI MP DA 4 A 32 BIT	55.000	
013 H	ASSEMBLER 6502	8.500	
016 H	ASSEMBLER Z80	8.500	
021 H	ASSEMBLER 68000	8.500	
025 H	ASSEMBLER 8086-8088	8.500	
029 H	ASSEMBLER 80286	8.500	
GE 567	80286 ARCHITETTURA E PROGRAMMAZIONE	58.000	
GY 603	80386 ARCHITETTURA E PROGRAMMAZIONE	37.000	

AUTOMAZIONE

CODICE	TITOLO	PREZZO	
208 A	CONTROLLORI PROGRAMMABILI	24.000	
616 P	CONTROLLO AUTOMATICO DEI SISTEMI	29.500	
GES251	STRUTTURA E FUNZIONAMENTO DEI CONTROLLI NUMERICI	29.000	
GES252	CONTROLLI NUMERICI: PROGRAMMAZIONE E APPLICAZIONI	28.000	
G 399	30 APPLICAZIONI DI CAD	29.000	
G 401	CAD/CAM & ROBOTICA	28.000	
CI 414	DAL CHIP ALLA ROBOTICA	15.000	
GE 547	LA PROGETTAZIONE AUTOMATICA	32.000	
GE 564	ROBOTICA - Fondamenti e applicazioni	38.000	

DIZIONARI ENCICLOPEDICI

CODICE	TITOLO	PREZZO	
DS 498	FISICA	14.000	
DS 499	MATEMATICA	14.000	
DS 522	GEOLOGIA	14.000	
DS 524	ELETTRONICA	14.000	
DS 525	ASTRONOMIA	14.000	
DS 526	CHIMICA	14.000	
DS 527	RAGIONERIA GENERALE	14.000	
DS 528	RAGIONERIA APPLICATA	14.000	
DS 529	BIOLOGIA	14.000	
DS 530	MECCANICA	14.000	
DS 531	INFORMATICA	14.000	

ARGOMENTI VARI

CODICE	TITOLO	PREZZO	
704 D	MANUALE PRATICO DI REGISTRAZIONE	10.000	
706 A	COMUNICAZIONI RADIO IN MARE	18.000	
800 H	FENDER, STORIA DI UN MITO	28.000	

SOFTWARE E MANAGEMENT TOOLS

CODICE	TITOLO	PREZZO	
CZ 469	GRAFIX - DISEGNARE CON IL PC	50.000	F
TP 606	CORSO AUTOISTRUZIONE LOTUS 1-2-3 (VERS. ITALIANA) F - MS DOS	90.000	F
TY 605	CORSO AUTOISTRUZIONE SUL SISTEMA MS DOS - FLOPPY	50.000	F
TY 640	TURBO PASCAL - LIBRERIA DI PROGRAMMI	40.000	F

CODICE	TITOLO	PREZZO	
TP 643	CORSO AUTOISTRUZIONE LOTUS 1-2-3 (INGLESE)	90.000	F
TP 608	BUDGET STRATEGICO (LOTUS 1-2-3)	100.000	F
TP 614	GESTIONE DELLE COMMESSE DI PRODUZIONE	100.000	F
TP 623	CONTROLLO DELLE VENDITE (CON MULTIPLAN)	100.000	F
TP 625	GESTIONE DEL PERSONALE (LOTUS 1-2-3)	100.000	F
TP 677	GESTIONE DELLE COMMESSE CON MULTIPLAN 2.0	100.000	F
TP 673	PREVENTIVO E CONSUNTIVO DEI COSTI - CON LOTUS 1-2-3 VERS. 2 E MULTIPLAN 2.0	100.000	F
TP 660	1-2-3 LIBRERIA DI MACRO	60.000	F
TY 691	SUPER SCREEN - UTILITY PER I PROGRAMMATORI	50.000	F
TY 690	PC DOCTOR UTILITY - RECOVERING DEI FILE	60.000	F
TP 644	STATISTICA A UNA E DUE DIMENSIONI	100.000	F
TP 681	ANALISI ABC CON LOTUS 1-2-3	100.000	F
TP 669	GESTIONE DELLE COMMESSE CON dBASE III PLUS	100.000	F
	MARKETING & MANAGEMENT		
M 648	PROBLEMI DI MARKETING	45.000	
M 649	DISTINTA BASE	23.000	
M 650	TECNICHE DI ANALISI FINANZIARIA	52.000	
	NOVITÀ SETTEMBRE '88		
M 647	RICERCHE DI MERCATO	72.000	
PP 641	AUTOCAD - Il grande manuale	55.000	
PP 728	VENTURA - Realizzazione e utilizzo dei fogli stile	42.000	F
PP 741	WORD versione 3 e 4	59.000	F
R 736	INSIDE PC IBM	63.000	F
R 734	MANUALE DEL DOS	55.000	
BE 721	MANUALE PRATICO DI ELETTRONICA DIGITALE	26.000	
BE 684	IL MANUALE DEI CMOS	35.000	

F = libro con floppy
C = libro con cassette

Per le vostre ordinazioni per corrispondenza utilizzate l'apposita cedola inserita in questa rivista.

* L'Editore si riserva di modificare i prezzi di copertina in qualsiasi momento.

J0008/C

È JACKSON

J0053

INFORMATICA PROFESSIONISTI

R. Farabone/L. Pinotti

FONDAMENTI DI COMMON LISP

Pag. 320 Lire 40.000
Cod. GY634

Il testo si pone come tramite indispensabile per il neofita che possieda solo conoscenze teoriche sulle problematiche, guidandolo con gradualità e numerose esemplificazioni, ad una adeguata e corretta padronanza del linguaggio.

INFORMATICA PROFESSIONALE

C.E. Panzalis

WORD

pp. 90 Lire 14.500
Cod. 050T

Una guida per una veloce consultazione ed un approccio dinamico al word processing che consente di comporre, organizzare, impaginare e quindi stampare un documento con qualità professionale.

ELETTRONICA CONSUMER

H.L. Davidson

MULTIMETRI DIGITALI

pp. 308 Lire 42.000
Cod. BE619

Indirizzato a tutte quelle persone, principianti o professionisti, interessate alla individuazione dei guasti e alla riparazione di apparecchi elettronici commerciali, mediante l'uso dei multimetri digitali.

H.W. Buchsbaum/R.J. Prestopnik

ENCICLOPEDIA DEI CIRCUITI INTEGRATI

pp. 536 Lire 60.000
Cod. BE639

Vengono fornite le principali informazioni su quasi 250 circuiti integrati, dai più semplici ai microprocessori a 32 bit.

ELETTRONICA PROFESSIONALE

Sybil P. Parker

ENCICLOPEDIA MONOGRAFICA DI ELETTRONICA E INFORMATICA

Vol. I
pp. 608 Lire 58.000
Cod. GE688

Vol. II
pp. 612 Lire 58.000
Cod. GE689

La sola enciclopedia in grado di fornire una panoramica completa ed approfondita dell'elettronica e dell'informatica, con articoli su argomenti di primario interesse ed estrema attualità.

IL TUO LIBRO

SERVIZIO LETTORI

IL GRUPPO EDITORIALE JACKSON PROMUOVE OGNI GIORNO NUOVE INIZIATIVE PER FACILITARE IL CONTINUO DIALOGO CON I PROPRI LETTORI. NATURALMENTE È IMPORTANTE CHE QUESTO SCAMBIO DI INFORMAZIONI SIA RESO IL PIÙ POSSIBILE AUTOMATICO E CHE I SUOI TEMPI SIANO SEMPRE PIÙ RISTRETTI. È CON QUESTO INTENTO CHE NASCE IL SERVIZIO LETTORI JACKSON, ORGANIZZATO IN MODO DA SODDISFARE OGNI ESIGENZA, SECONDO UN SISTEMA DI CEDOLE PRECONFIGURATE, DA INVIARE AL NOSTRO SERVIZIO MARKETING. ANZITUTTO, IL SERVIZIO LETTORI JACKSON CONSENTE DI SOTTOSCRIVERE ABBONAMENTI O ORDINARE LIBRI E GRANDI OPERE UTILIZZANDO LE CEDOLE QUI A FIANCO, SCEGLIENDO LA MODALITÀ DI PAGAMENTO PREFERITA. UN ESTRATTO CONDENSATO DEL CATALOGO LIBRI E GRANDI OPERE JACKSON È PUBBLICATO NELLE ULTIME PAGINE DI QUESTA RIVISTA; IL CATALOGO COMPLETO PUÒ ESSERE

COMUNQUE ORDINATO, UTILIZZANDO LA CEDOLA NUMERO 3: INFORMAZIONI & AGGIORNAMENTI. QUEST'ULTIMA È LA PIÙ IMPORTANTE E PERMETTE AL LETTORE DI RICEVERE, DIRETTAMENTE A CASA PROPRIA, TUTTE LE INFORMAZIONI SULLE INIZIATIVE JACKSON CHE LO INTERESSANO: CATALOGHI, LIBRI, CAMPAGNA ABBONAMENTI CORSI DELLA DIVISIONE FORMAZIONE E PRODOTTI PER LA DIDATTICA JACKSON S.A.T.A., COPIE OMAGGIO DI RIVISTE E FASCICOLI DI GRANDI OPERE. QUESTO SERVIZIO CONSENTE, OLTRE CHE DI RIMANERE AGGIORNATI, ANCHE DI AGGIORNARE I COLLEGHI E GLI AMICI, POICHÈ LA CEDOLA È STUDIATA ANCHE CON QUESTO INTENTO. NON PIÙ TELEFONATE E LETTERE: DA OGGI È SUFFICIENTE SPEDIRE L'APPOSITO TAGLIANDO, PER OTTENERE IN BREVISSIMO TEMPO IL MATERIALE DESIDERATO.

CEDOLA ABBONAMENTO RIVISTE JACKSON

Se desiderate sottoscrivere abbonamenti alle riviste Jackson, utilizzate questa cartolina. Gli abbonati Jackson possono contare su un duplice risparmio (una tariffa privilegiata e la garanzia del prezzo bloccato per la durata del proprio abbonamento) e hanno diritto a uno sconto negli acquisti di libri. **Ritagliate e spedite, riportando sulla busta l'indirizzo esatto del Gruppo Editoriale Jackson.**

RITAGLIATE E SPEDITE IN BUSTA CHIUSA

MITTENTE

COGNOME ______

NOME ______

VIA E NUMERO ______

CAP ______ CITTÀ ______

PROV. ______ TEL. (______) ______

GRUPPO EDITORIALE **JACKSON**
Via Rosellini, 12
20124 Milano

CEDOLA COMMISSIONE LIBRI E GRANDI OPERE

Se desiderate ordinare libri o "Grandi Opere Jackson", utilizzate questa cedola. Compilate gli appositi spazi precisando anche il tipo di pagamento scelto, il vostro nome, cognome e indirizzo. **Ritagliate e spedite, riportando sulla busta l'indirizzo esatto del Gruppo Editoriale Jackson.**

RITAGLIATE E SPEDITE IN BUSTA CHIUSA

MITTENTE

COGNOME ______

NOME ______

VIA E NUMERO ______

CAP ______ CITTÀ ______

PROV. ______ TEL. (______) ______

GRUPPO EDITORIALE **JACKSON**
Via Rosellini, 12
20124 Milano

CEDOLA INFORMAZIONI E AGGIORNAMENTI

Se desiderate ricevere rapidamente informazioni sui prodotti e attività del Gruppo Editoriale Jackson o acquistare, con formula rateale a sole L. 25.000 mensili e un anticipo di L. 45.000 una "Grande Opera Jackson", barrate le caselle della cedola che vi interessano. **Ritagliate e spedite, riportando sulla busta l'indirizzo esatto del Gruppo Editoriale Jackson.**

RITAGLIATE E SPEDITE IN BUSTA CHIUSA

MITTENTE

COGNOME ______

NOME ______

VIA E NUMERO ______

CAP ______ CITTÀ ______

PROV. ______ TEL. (______) ______

GRUPPO EDITORIALE **JACKSON**
Via Rosellini, 12
20124 Milano

SÌ DESIDERO ABBONARMI ALLE SEGUENTI RIVISTE JACKSON:

Rivista	Numeri	Prezzo
☐ INDUSTRIA OGGI	numeri 10	L. 41.000
☐ ELETTRONICA OGGI	numeri 20	L. 79.000
☐ AUTOMAZIONE OGGI	numeri 20	L. 78.000
☐ EO NEWS	numeri 40	L. 79.500
☐ WATT	numeri 11	L. 35.500
☐ TRASMISSIONE DATI E TELECOMUNICAZIONI	numeri 11	L. 44.000
☐ VIDEOTEL MAGAZINE	numeri 6	L. 20.000
☐ INFORMATICA OGGI	numeri 11	L. 40.000
☐ INFORMATICA OGGI SETTIMANALE	numeri 40	L. 80.000
☐ LAB NEW	numeri 8	L. 36.000
☐ COMPUTER GRAFICA & APPLICAZIONI	numeri 6	L. 30.000
☐ PC WORLD MAGAZINE	numeri 11	L. 44.000
☐ PCW Magazine + PC Floppy	numeri 11	L. 105.000
☐ BIT	numeri 11	L. 43.000
☐ SUPERCOMMODORE 64 & 128 (con cassetta)	numeri 11	L. 66.000
☐ SUPERCOMMODORE 64 & 128 (con disco)	numeri 11	L. 105.000
☐ COMMODORE PROFESSIONAL	numeri 10	L. 48.000
☐ OLIVETTI PRODEST USER	numeri 6	L. 20.000
☐ LA RIVISTA DI ATARI	numeri 6	L. 24.000
☐ COMPUSCUOLA	numeri 10	L. 32.000
☐ FARE ELETTRONICA	numeri 12	L. 39.000
☐ STRUMENTI MUSICALI	numeri 11	L. 35.000
☐ NAUTICAL QUARTERLY	numeri 4	L. 70.000
☐ AUTOMOBILE QUARTERLY	numeri 4	L. 69.500

☐ L'abbonamento dovrà decorrere dal mese di ______

MODALITÀ DI PAGAMENTO

Per il pagamento ☐ Allego assegno n. ______ di L. ______

Banca ______

☐ Ho effettuato versamento di L. ______ sul c/c postale n. 11666203 intestato a Gruppo Editoriale Jackson - Milano e allego fotocopia della ricevuta.

☐ Ho effettuato versamento di L. ______ tramite vaglia postale o telegrafico e allego fotocopia della ricevuta.

☐ Vi autorizzo ad addebitare l'importo di L. ______ sulla carta di credito.

☐ VISA ☐ AMERICAN EXPRESS ☐ DINERS CLUB ☐ CARTA SI

N. ______ Data di scadenza ______

Data ______ Firma ______

N.B. Per abbonamenti all'estero le tariffe dovranno essere raddoppiate. Non è prevista la spedizione via aerea.

SÌ INVIATEMI LE SEGUENTI "GRANDI OPERE JACKSON"

Codice	Q.tà	Prezzo	Codice	Q.tà	Prezzo

Ordine minimo L. 30.000 + L. 3.500 per contributo fisso spese di spedizione

☐ **Sono abbonato** alla seguente rivista Jackson: ______
e ho quindi diritto allo sconto del **10%.**

SÌ INVIATEMI I VOLUMI SOTTOELENCATI:

Volume	Prezzo
☐ **EI - Enciclopedia di Elettronica e Informatica** 10 volumi	L. 476.000
☐ **SOFTWARE** 5 volumi	L. 236.000
☐ **DEI - Dizionario di Elettronica e Informatica** 10 volumi	L. 276.000
☐ **Enciclopedia Monografica di Elettronica e Informatica** 2 volumi	L. 116.000
☐ **ABC Personal Computer** 4 volumi	L. 136.000
☐ **VIDEOBASIC MSX** 20 lezioni + 20 cassette	L. 176.000
☐ **VIDEOBASIC SPECTRUM** 20 lezioni + 20 cassette	L. 176.000
☐ **VIDEOBASIC** C64/C128/64PC 20 lezioni + 20 cassette	L. 176.000
☐ **VIDEOBASIC** C64/C128/64PC 20 lezioni + 10 floppy	L. 176.000
☐ **VIDEOBASIC** C16/PUS4 20 lezioni + 20 cassette	L. 176.000
☐ **VIDEOBASIC** VIC20 20 lezioni + 20 cassette	L. 176.000
☐ **CORSO DI GRAFICA** C64/C128/64PC 10 lez. + 10 cass.	L. 96.000
☐ **A SCUOLA DI SCACCHI** C64/C128/64PC 10 lez. + 10 cass.	L. 96.000
☐ **7 NOTE BIT** C64/C128/64PC 15 lezioni + 15 cassette	L. 156.000
☐ **LABORATORIO DI ELETTRONICA** 5 vol. (disp. da giugno 88)	L. 236.000
☐ **BYTES** 6 volumi (disp. da giugno 1988)	L. 276.000

☐ Allego assegno n. ______ di L. ______
della Banca ______

☐ Ho effettuato il pagamento di L. ______ a mezzo:
☐ vaglia postale ☐ vaglia telegrafico ☐ versamento sul c/c postale n. 11666203 intestato a Gruppo Editoriale Jackson SpA Milano e allego fotocopia della ricevuta.

☐ Pagherò al postino l'importo di L. ______ al ricevimento dell'opera.

☐ Vi autorizzo ad addebitare l'importo di L. ______ sulla carta di credito: ☐ Visa ☐ American Express ☐ Diners Club ☐ Carta SI
conto n. ______ data di scadenza ______

☐ Richiedo l'emissione della fattura (formula riservata alle aziende) e comunico il numero di Partita IVA ______

DATA ______ FIRMA ______

CEDOLA INFORMAZIONI E AGGIORNAMENTI

☐ Desidero riceve informazioni sulle **riviste** JACKSON
☐ Desidero ricevere una copia saggio della **rivista** JACKSON ______
(indicare titolo)
☐ Desidero ricevere il **catalogo libri tecnici** JACKSON
☐ Desidero ricevere il **catalogo libri scolastici** JACKSON
☐ Desidero ricevere il **catalogo grandi opere** JACKSON e una copia saggio ______
(indicare titolo)
☐ Desidero ricevere informazioni per l'acquisto **in forma rateale delle Grandi Opere** JACSKON e una copia saggio ______
(indicare titolo)
☐ Desidero ricevere il **catalogo occasioni** JACKSON

Desidero ricevere maggiori informazioni sulla **Divisione Formazione e Prodotti per la Didattica**
☐ Area ELETTRONICA E MICROPROCESSORI
☐ Area TELECOMUNICAZIONI E TELEMATICA
☐ Area AUTOMAZIONE INDUSTRIALE E ROBOTICA
☐ Area INFORMATICA E INTELLIGENZA ARTIFICIALE
☐ Area ALTE TECNOLOGIE SPECIALI
☐ Laboratorio di INFORMATICA
☐ Laboratorio di TELEMATICA
☐ Laboratorio di OFFICE AUTOMATION
☐ Laboratorio di ELETTRONICA
☐ Laboratorio di MICROPROCESSORI
☐ Laboratorio di AUTOMAZIONE INDUSTRIALE